سلام دنیا

```
DELIMITER $$
CREATE FUNCTION hello_world()
RETURNS TEXT COMMENT 'Hello World'
BEGIN
RETURN 'Hello World';
END;
$$
DELIMITER ;

SELECT hello_world();
```

ماهنامه تخصصی نرم افزارهای آزاد/متن باز | شماره هفتم | شهریور ماه ۱۳۹۴ | ۱۰۲ صفحه

پرونده:

آشنایی با مدل‌های مدیریت داده‌ها

با انواع پایگاه‌داده‌های پرکاربرد آشنا شوید

روزی که مدیر سیستم‌ها بر زمین حکم راندند

زندگی با شیطانک قرمز

متن‌باز شدن دیگر اختیاری نیست، حتی برای اپل

۷ کتابی که هر مدیر جامعه‌ای باید مطالعه کند

تدابیری که دولت‌ها برای ترویج نرم‌افزار آزاد باید به کار گیرند

اهمیت چرخش آزاد اطلاعات دولتی

صرفه‌جویی میلیون دلاری و دانش آزاد

واکنش به حملات سایبری مهم‌تر از جلوگیری است

Linux Professional Institute
Islamic Republic of Iran

دوره های حرفه ای لینوکس

در تنها مرکز تخصصی آموزش و آزمون بین المللی لینوکس ایران

فناوران آنیسا

- دوره آشنایی با لینوکس (Linux Essential)
- دوره مدیریت پایه لینوکس LPIC-۱
- دوره مدیریت پیشرفته لینوکس LPIC-۲
- دوره پیشرفته محیط های مختلط LPIC-۳۰۰ (LDAP, Samba)
- دوره پیشرفته امنیت در لینوکس LPIC-۳۰۳
- دوره پیشرفته مجازی سازی در لینوکس LPIC-۳۰۴
- دوره مقدماتی و پیشرفته Embedded Linux
- دوره پیشرفته مانیتورینگ لینوکس
- دوره مقدماتی و پیشرفته Astrisk VoIP
- دوره استادی اسکریپت نویسی (Mastering Linux Shell Scripting)
- دوره طراحی و برنامه نویسی وب (LAMP)
- برگزاری آزمون های بین المللی LPI
- و هم اکنون دوره تخصصی رایانش ابری **(Cloud Computing)**

نشانی ما: میدان آرژانتین، بلوار بیهقی، دوازدهم شرقی، پلاک ۶ - تلفن تماس: ۸۸۵۴۸۶۰۳ و ۸۸۵۴۸۳۶۰

www.anisa.co.ir

گروه پیشگامان
(تعاونی)

پیشنهاد شگفت انگیز

تا ۱۰ مگابیت | ۵۰ روز | ۵ گیگ

قیمت : ۵۰۰۰ تومان

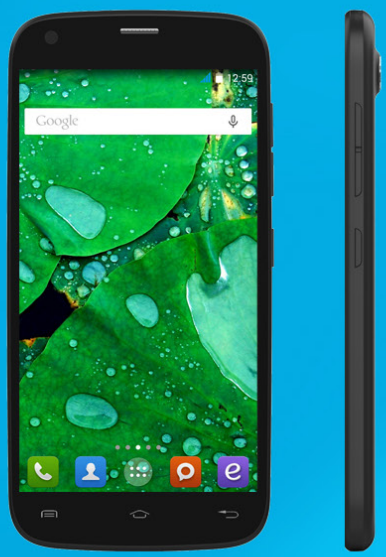
شماره تماس: ۲۳۵۴۵

www.Pishgamandsl.com

GLX

ارتباط همراه گویا اروند

Spring Pro



Quad Core 1.3 MTK6582

RAM 1 GB Built-in Storage 16 GB

IPS LCD 4.6" 540x960

8.0 Mega Pixel

Android 4.4

Dual SimCard

1,500 MA

Spider 1 Pro

Quad Core 1.3 MTK6582

RAM 2 GB Built-in Storage 16 GB

IPS LCD 5.0" 720x1280

13.0 Mega Pixel

Android 4.4

Dual SimCard

USB On-The-Go

2,000 MA

Crystal

Dual Core 1.3 MTK6582

RAM 512 MB Built-in Storage 4 GB

IPS LCD 7.0" 600x1024

2.0 Mega Pixel

Android 4.4

Dual SimCard

USB On-The-Go

3,000 MA

Spark Pro

Quad Core 1.3 MTK6582

RAM 1 GB Built-in Storage 4 GB

IPS LCD 5.0" 720x1280

8.0 Mega Pixel

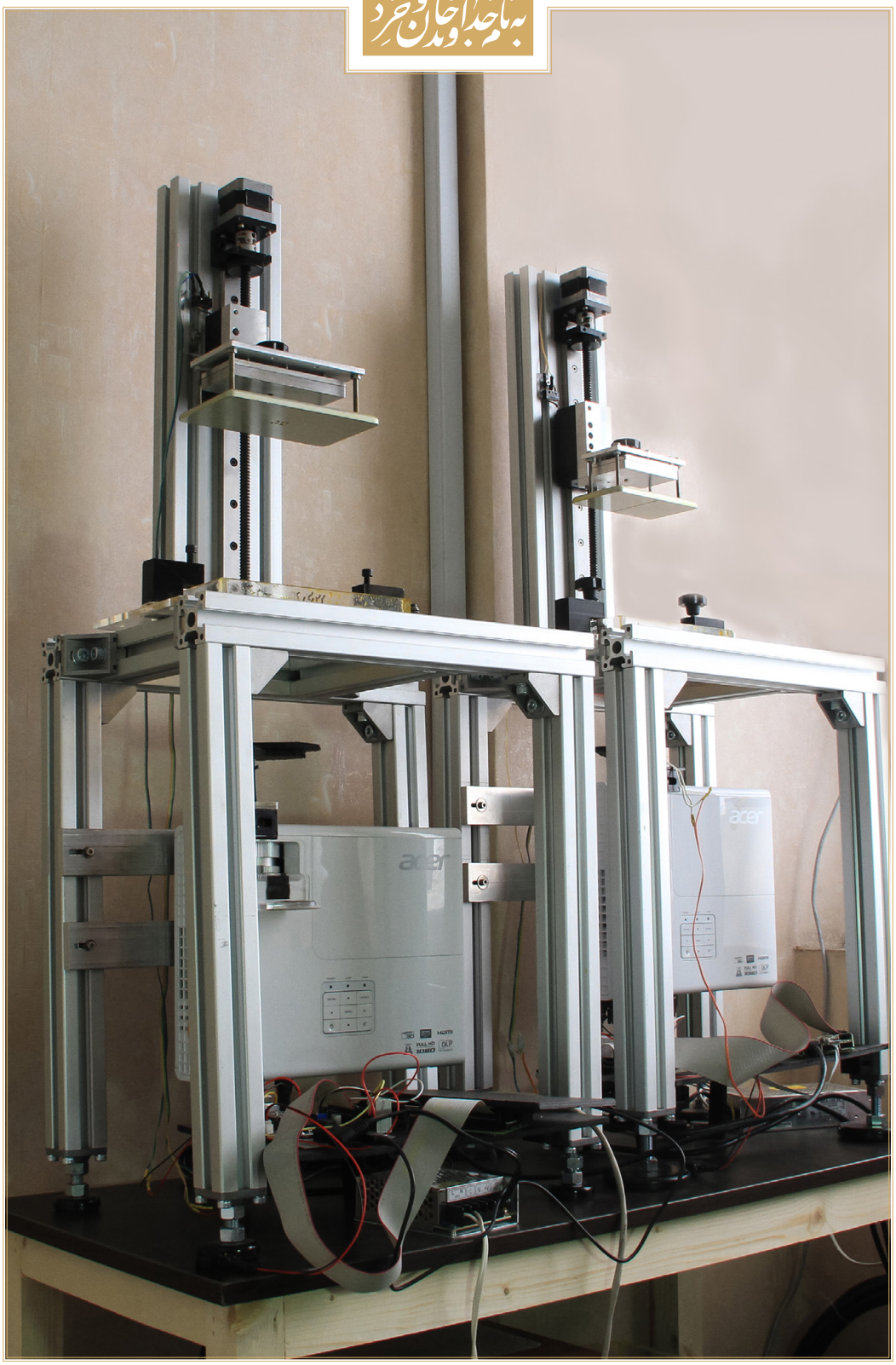
Android 4.4

Dual SimCard

USB On-The-Go

2,000 MA

به نام خداوند جان و خرد



چاپگر سه‌بعدی ایرانی ساخته شده بر اساس دانش آزاد

سلام دنیا

ماهنامه تخصصی نرم‌افزارهای آزاد/ متن‌باز

شماره هفتم | شهریور ماه ۱۳۹۴

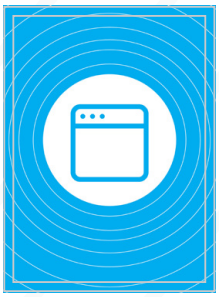
- صاحب امتیاز و مدیر مسئول:
  یاسر توکلی کرمانی
- سردبیر:
  محمد دماوندی ▪ eic@salam-donya.ir

شورای سردبیری:
- **محمد افاضاتی**
- **محمد درویش**
- **علی فارمد**
- **احسان کریم‌خانی**
- **محمد نبی‌زاده**

همکاران این شماره:
دانیال بهزادی ▪ **مهدی بیگی** ▪ **احسان ترک**
امیرحسین حسینی‌پژوه ▪ **سپهر حمزه‌لوی** ▪ **مهدی حمیدی**
مهدی خشنودی ▪ **محمد خلاقی** ▪ **حمیدرضا سلیمانی**
وحید سهرابلو ▪ **سهراب سهیلیان** ▪ **نیما صحرانشین سامانی**
مسعود صدرنژاد ▪ **سعید علیجانی** ▪ **سعیده علی‌محمدی**
جادی‌میرمیرانی

- مدیرهنری: **علیرضا بخشی**
- عکاس: **نگار مصطفوی** ▪ **محسن کرامت‌دهر**

نشانی: تهران ▪ فلکه دوم صادقیه ▪ ابتدای بلوار فردوس
پلاک ۱۴ ▪ واحد ۱ ▪ تلفن: ۴۴۰۰۰۷۵۱ (۰۲۱)
www.salam-donya.ir ▪ info@salam-donya.ir

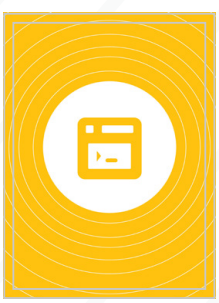


معرفی

کسب و کار

جامعه کاربری

پرونده



تخصصی

سخن نخست

محمد دماوندی
سردبیر

نرم‌افزارهای مختلف از مرورگر و دستیارهای نرم‌افزاری مانند کورتانا، سیری و مشابه آن‌ها گرفته تا سیستم‌عامل، از تلفن همراه، تبلت و ساعت هوشمند گرفته، حتی تا تلویزیون، همه و همه نظاره‌گر رفتار، اعمال و گفتار ما هستند و در آینده‌ای نه چندان دور، خودرو و حتی پیراهنی هم که به تن داریم به جمع این ابزار خواهند پیوست. بسیاری از این تجهیزات بدون آن‌که ما بخواهیم به بهانه‌های مختلف، از جمله کمک در بهبود برنامه و کارایی آن، همگام‌سازی دستگاه‌های مختلف و یا به منظور به اصطلاح تسهیل در امور روزمره، در تکاپوی جمع‌آوری اطلاعات ما هستند. شاید با استفاده از نرم‌افزارهای آزاد / متن‌باز بخشی از این تلاش‌ها را بتوانیم کنترل کنیم. اما وقتی که پای سخت‌افزار هم به میان می‌آید، دست‌هایمان کاملا بسته است. وقتی پی می‌بریم که «دوستان بزرگوار» از طریق پردازنده هم به جمع‌آوری داده‌های ما مشغول هستند، دیگر دلیل واهی تسهیل در امور روزمره کاملا رنگ می‌بازد و آسودگی خاطر برای لذت بردن از امور تسهیل شده، جای خود را به ترس و ناامنی می‌دهد.

از چند دهه گذشته تا همین چندسال پیش، جمع‌آوری داده‌ها یکی از دغدغه‌های اصلی بود. بیشتر سازمان‌ها هزینه زیادی برای دسترسی به اطلاعات کاربران صرف می‌کردند تا بدانند دیدگاه کاربرانشان چیست، بینندگان تلویزیونی از کدام برنامه رضایت بیشتری دارند، کاربران مختلف ابزارشان را با چه کاربردهای خلاقانه‌ای به کار می‌گیرند ، سیستم در حالت‌های خاص چه رفتاری از خود بروز می‌دهد و مواردی از این دست. امروزه با گسترش اینترنت و ابزارک‌های همراه این نیاز تا حد زیادی مرتفع شده و اکنون گرایش‌ها بیشتر به کاربرد داده‌ها سوق پیدا کرده و دیگر سازمان‌ها آن قدر اطلاعات دارند که دسترسی آسان، بهینه و ساخت‌یافته به آن‌ها اهمیت بیشتری پیدا کرده است.

پیدایش مدل‌های جدید پایگاه‌های داده، مفاهیم «هوش کسب و کار» را که سال‌ها پیش توسط ریچارد میلار ارایه شده بود، وارد دوره جدیدی می‌کند. آن‌قدر اطلاعات داریم که به سرعت از تعاریف می‌گذریم و بتوانیم شاهد ظهور حوزه‌های جدیدی در پردازش داده‌ها باشیم. هوش تجاری ابری، هوش تجاری موبایل و سایر موارد همه زاده دپوی انبوه اطلاعات است.

در گذشته گزینه‌های محدودی برای ذخیره و بازیابی داده‌ها داشتیم و امروز شاهد پیدایش پایگاه‌های داده خاص‌منظوره هستیم. به لطف عرضه محصولات آزاد / متن‌باز نیز می‌توانیم برای نیازهای خود به جای اختراع مجدد چرخ بر بهبود چرخ‌های موجود تمرکز کنیم. حتی برای بخش‌های مختلف یک سامانه نیز می‌توانیم فناوری‌های مختلف را ترکیب و خدمات کارآمدتری ارایه کنیم

براساس آمار ارایه شده توسط «برادر بزرگ»، جستجوی عباراتی همچون Big Data در ده سال اخیر بیش از ۱۷ برابر رشد داشته است.

تمامی این نکات ما را به درک این نکته رهنمون می‌سازد که بر لبه عصر مدیریت اطلاعات و پایگاه داده‌ها ایستاده‌ایم و باید منتظر تغییر نگرش، ابزارها و کاربردهای مختلف داده‌ها و اطلاعات باشیم. تبدیل داده‌ها به اطلاعات و اطلاعات به دانش در چشم‌برهم‌زدنی صورت می‌پذیرد. مدیران به اهمیت این موضوع پی‌برده‌اند و ذائقه‌شان تغییر یافته است.

استفاده از یک نمودار تحلیلی ساده، بدون چالش، اولویت‌ها را مشخص می‌کند؛ هرچند تهیه همین نمودارها نیز مستلزم دانش، مهارت و تیزبینی است.

چگونه است که شبکه اجتماعی فیس‌بوک با بیش از ۱،۵۰۰،۰۰۰،۰۰۰ کاربر فعال در ماه، بدون اختلال محسوس، بر روی پایگاه‌های داده مای‌اس‌کیوال استوار است، ولی در کشور عزیزمان، هنگام اعلام نتایج کنکور، خواب بر مدیران سیستم حرام می‌شود!؟

بنابراین در وهله اول باید با دوراندیشی و ترسیم چشم‌اندازی مشخص برای نحوه مدیریت و کاربرد داده‌هایمان، بستر مناسب را انتخاب کنیم و بر اساس نیازمندی‌ها، قرارگاه اطلاعات را طوری مشخص کنیم که با گذر زمان به آرامگاه اطلاعات تبدیل نشود. انتخابی صحیح، مستلزم شناختی دقیق است. بدون داشتن درک کاملی از نیازمندی‌های فعلی و آتی و بررسی جزییات پایگاه‌داده‌ها، شناخت نواقص و ویژگی‌های برتر هر یک، هر انتخابی دچار اختلال می‌شود.

در این شماره به بررسی مدل‌های مختلف مدیریت داده‌ها و ابزارهای آزاد / متن‌باز پرداخته‌ایم تا انگیزه لازم را جهت کاربرد پایگاه‌های داده قدرتمند آزاد برای متخصصان و مدیران خلاق ایجاد کنیم و این پیام مهم را به گوش کاربران متعصب پایگاه‌های داده انحصاری برسانیم که اگر حجم انبوهی از داده‌ها را در اختیار دارید و مدیریت می‌کنید، اگر تعداد تراکنش‌های بالایی برای خود متصور هستید، آگاه باشید که استفاده از ابزار انحصاری و گران‌قیمت، الزاما مدیریت بهتر را به ارمغان نمی‌آورد.

http://slmd.ir/jx

جشن تولد ۳۰ سالگی بنیاد نرم‌افزار آزاد

بنیاد نرم افزار آزاد، دعوت باز خود را برای شرکت در جشن تولدش، اعلام کرد. این جشن در غروب شنبه، یازدهمین روز ماه مهر ۱۳۹۴ در بوستون برگزار می‌شود و از حضور رئیس و موسس بنیاد نرم افزار آزاد، استالمن بهره‌مند خواهد شد. این جشن نشان دهنده اوج سی سالگی است که دربردارنده بزرگترین کنفرانس «لیبره‌پلنت» تا کنون و یک مقاله درباره گنو در نیویورکر است. علاوه بر این‌ها، بنیاد یک کنفرانس کوچک را نیز در نظر گرفته است که زمانش ۱۱ مهر ۹۴ و در طول روز است. در این کنفرانس، جامعه نرم‌افزار آزاد درس‌هایی که در این سی سال به دست آورده است را به اشترک می‌گذارند و برای آینده برنامه‌ریزی می‌کنند. همچنین احتمال برگزاری یک مراسم شام برای جمع آوری کمک مالی در جمعه ۱۰ مهر ماه نیز وجود دارد. به عنوان رویدادهای موازی، در سراسر جهان نیز حامیان، علاقه‌مندی خود را به برگزاری یک رویداد محلی برای این تولد بیان کرده‌اند. بنیاد از پوشش دادن این رویدادها بر روی بلاگ خود یا اتصال آن‌ها به جشن اصلی خوشحال خواهد شد و استقبال می‌کند. اگر علاقه‌مند به برگزاری رویدادی مشابه در منطقه خود هستید با campaigns@fsf.org تماس بگیرید.■

http://sfd.fsug.ir/1394/

روز آزادی نرم افزار

جشنواره روز آزادی نرم‌افزار تهران، مورخ ۹ مهر ۱۳۹۴ از ساعت ۹ تا ۱۶ در دانشگاه صنعتی شریف، در دو بخش عمومی و کارگاه‌ها و با محوریت معرفی، ترویج، افزایش کاربرد و استفاده، جذب مشارکت و حمایت جامعه و تولید در زمینه نرم‌افزار آزاد برگزار خواهد شد. علاقه‌مندان می‌توانند مقالات و موضوعات پیشنهادی خود را جهت ارائه در بخش کنفرانس، یا کارگاه‌ها تا تاریخ ۲۰ شهریور از طریق صفحه ارسال مقاله به دبیرخانه جشنواره ارسال کنند، تا پس از داوری، مقالات برگزیده برای ارائه در روز جشنواره اعلام شوند. کلیه موضوعات تخصصی یا عمومی در حوزه نرم‌افزار آزاد قابل پذیرش است. برای اطلاعات بیشتر درباره روش داوری، معیارهای انتخاب مقالات و نحوه تنظیم و ارسال مقاله سایت جشنواره را مطالعه کنید.■

http://slmd.ir/kf

تهدیدی برای فایل‌ها با فرد میان ابری

با استفاده روشی به نام «فرد میان ابری»، مهاجمان می‌توانند به واسطه سرویس همگام‌سازی ابری، به فایل‌های شما در گوگل‌درایو، دراپباکس و وان‌درایو بدون زمر، دسترسی پیدا کنند.

براساس پژوهش‌های اخیر که در کنفراکس بلک‌هت در لاس‌وگاس در ۱۴ مرداد ۹۴ مطرح شده است، نحوه دسترسی به فضای سرویس‌های ذخیره‌سازی ابری، به نمایش درآمده است.■

http://slmd.ir/gz

این دستگاه افکارتان را تایپ می‌کند

در مقاله‌ای از نشریه مادربرد، به تحقیقات انجام شده در آزمایشگاه شالک اشاره شده است که در آن محققین به تازگی نشان داده‌اند می‌توان افکار یک فرد را با چیزی که آن را «مغز به نوشتار» می‌نامند تبدیل به عباراتی خوانا کرد. جیسون کبلر می‌نویسد «این تازه روزهای آغازین این فناوری است، الکترودها دقیقا باید بر روی مغز جای‌گذاری شوند و واژه نامه عبارات نیز محدود است. با این حال امواج مغزی ساطع شده از الگوی فکری با احتمالی بیشتر از شانس به متن تبدیل می‌شوند.»■

DuckDuckGo

All Friends Family Acquaintances Following

http://slmd.ir/ku

آزادرسانی سامانه تَرگمان صورت گرفت

سامانه ترجمه ماشینی پژوهشگاه ارتباطات و فناوری اطلاعات (مرکز تحقیقات مخابرات ایران) موسوم به «ترگمان»، با امکان ترجمه روزانه بیش از یک میلیون کلمه، آزاد رسانی (متن باز) شد.

به گزارش روابط عمومی پژوهشگاه ارتباطات و فناوری اطلاعات، این سامانه که در راستای حمایت از توسعه صنعت ترجمه ماشینی طراحی و فعال شده است، حاصل تلاش چند ساله پژوهشگاه با همکاری دانشگاه صنعتی امیرکبیر است. این گزارش حاکی است، کدها و مستندات این محصول در سامانه اشتراک فایل OpenGit.ir که شامل ابزارهای لازم جهت راه‌اندازی یک مترجم ماشینی آماری دوسویه انگلیسی/فارسی است، بارگذاری شده است. خدمت ترجمه ماشینی مبتنی بر محصول ترگمان در نشانی اینترنتی targoman.ir ارائه می‌شود و دسترسی به کدها و مستندات سامانه ترجمه ماشینی پس از ورود به سامانه آزادرسانی فوق و مراجعه به قسمت مترجم ماشینی ترگمان و ثبت‌نام مطابق با ضوابط مربوطه امکان‌پذیر است.

گفتنی است سامانه ترجمه ماشینی ترگمان امکان راهنمایی، پاسخ‌گویی به پرسش کاربران و ارایه اطلاعات تکمیلی را دارا است.

http://slmd.ir/kv

آخرین وضعیت شبکه ملی اطلاعات تشریح شد

به گفته علی‌اصغر انصاری بخش اختصاصی شبکه ملی اطلاعات، ارتباط دستگاه‌های اجرایی با زیرمجموعه است و این ارتباطات در کشور ۸۰ درصد برقرار شده است. معاون توسعه و مدیریت شبکه ملی اطلاعات در همایش معرفی شبکه ملی اطلاعات و خدمات مرکز داده استان کرمان اظهار کرد: تحقق شبکه ملی اطلاعات به عنوان زیرساخت ارتباط فضای مجازی کشور نیازمند شبکه‌ای متشکل از زیرساخت‌های ارتباطی با مدیریت مستقل کاملا داخلی، شبکه‌ای با قابلیت عرضه انواع خدمات امن به کلیه کاربران و شبکه ای پرظرفیت، پهن باند و با تعرفه رقابتی است.

همچنین وی افزود که در حال حاضر قریب به ۱۰ میلیون پورت پرسرعت ثابت در کشور راه‌اندازی و به بهره برداری رسیده است. سیاست‌های ابلاغی مقام معظم رهبری در برنامه ششم توسعه در حوزه فناوری اطلاعات، نمونه جدیدی است و تاکنون به این صراحت نیامده است. وی اظهار داشت: دربند ۳۲ به کسب جایگاه برتر منطقه در توسعه دولت الکترونیک در بستر شبکه ملی اطلاعات اشاره شده و در این زمینه ماموریت‌ها کاملا تفکیک شده است.

http://slmd.ir/kw

اعطای بیش از ۲۶ میلیارد تومان تسهیلات

در قالب فراخوان اول اعطای تسهیلات از محل وجوه اداره شده، اعطای بیش از ۲۶ میلیارد تومان تسهیلات به ۷۳ شرکت خصوصی فعال در حوزه ارتباطات و فناوری اطلاعات به تائید کمیته وجوه اداره شده وزارت ICT رسید. این را دکتر صمیمی مدیرکل توسعه فناوری و دبیر وجوه اداره شده وزارت ارتباطات و فناوری اطلاعات، با اعلام پایان فرآیند ارزیابی و اعطای تسهیلات در چارچوب فراخوان سال ۹۳، اعلام کرد.

دکتر صمیمی با اشاره به استفاده حداکثری از بضاعت فنی موجود در بخش خصوصی کشور برای ارزیابی طرح های دریافتی، افزود: با پیشنهاد کمیته وجوه اداره شده و تائید وزیر ارتباطات، از این پس نمایندگان نظام صنفی رایانه ای و سندیکای صنعت مخابرات به صورت رسمی عضو کمیته‌های تخصصی وجوه اداره شده خواهند شد و تعامل با این تشکل ها برای معرفی نمایندگان مربوطه در حال انجام است.

http://slmd.ir/ggc5migrate

به‌روز رسانی بزرگ

نگارش ۵ جی‌سی‌سی در اردیبهشت ماه عرضه شد و اکنون توزیع‌های دبیان و اوبونتو در فکر به‌روز کردن تمام بسته‌های نرم‌افزاری خود هستند تا در نهایت نرم افزارهایشان با این نگارش جدید، کامپایل شود.

جی‌سی‌سی، مخفف GNU Compiler Collection به معنی مجموعه کامپایلرهای گنو، کدهای مبدأ نوشته شده توسط برنامه‌نویسان را به کتاب‌خانه‌ها و کدهای اجرایی تبدیل می‌کند. از این مجموعه کامپایلر برای ساخت هر چیزی، از لینوکس گرفته تا نرم‌افزارهای کاربردی، استفاده می‌شود. پنجمین نگارش جی‌سی‌سی تغییرات اساسی بیشتری را نسبت به نگارش‌های پیشین معرفی کرده است و نخستین باری است که به طور کامل از نگارش آخر زبان برنامه نویسی سی‌پلاس‌پلاس پشتیبانی می‌کند. برای به‌روز کردن بسته‌ها باید تمام آن‌ها را از ابتدا کامپایل کرد.

http://slmd.ir/kb

کشف آسیب‌پذیری ۱۸ ساله در پردازنده‌ها

آسیب‌پذیری ۱۸ ساله‌ای در پردازنده‌های x86 اینتل کشف شده است که درها را به روی روت‌کیت‌ها باز می‌کند. پژوهشگر امنیتی، کریستفر دوماس، روشی یافته تا بتواند روت‌کیت را بر روی میان‌افزار رایانه نصب کند. این روش از نقضی در طراحی تراشه‌هایی که از سال ۱۹۹۷ میلادی ساخته شده است، استفاده می‌کند. این روت‌کیت حالت مدیریت سیستم پردازنده (System Management Mode) را تحت تاثیر قرار می‌دهد و امکان پاک کردن UEFI یا حتی تاثیر مجدد پس از نصب سیستم‌عامل را می‌دهد. راه‌کارهای حفاظتی مانند Secure Boot کمکی نمی‌کند چرا که با اعتماد کامل بر SMM، امن شده‌اند. این روش تنها بر روی پردازنده‌های x86 اینتل آزموده شده اما به گفته دوماس، پردازنده‌های AMD هم باید دارای چنین آسیب‌پذیری‌ای باشند.

http://slmd.ir/kx

لیبره‌آفیس ۵ منتشر شد

نسخه جدیدی از نرم‌افزار لیبره‌آفیس ویرایش تازه «Fresh» منتشر شد. این ویرایش معمولا دارای جدیدترین ویژگی‌های موجود در این مجموعه اداری است و از تمامی ویژگی‌های تازه برخوردار است. با وجود این، کاربرانی که به پایداری بیشتر اهمیت می‌دهند نیز قادر خواهند بود از ویرایش دیگری با نام «Still» استفاده کنند که نسخه گذشته از مجموعه اداری لیبره‌آفیس به همراه رفع‌باگ تا چندین ماه را شامل می‌شود که به جای تمرکز بر افزودن موارد و ویژگی‌های جدید بر رفع نواقص تکیه دارد. لیبره‌آفیس جدید از ویژگی‌های متنوعی برخوردار بوده‌است که این نسخه را می‌توان بزرگ‌ترین تغییرات در این مجموعه نرم‌افزار اداری از بدو تاسیس بنیاد اسناد و انشعاب این مجموعه از اپن‌آفیس به حساب آورد.

http://slmd.ir/kt

مرحله نخست فراخوان جویشگر بومی

در مرحله نخست فراخوان جویشگر بومی، ۷۹ پیشنهاد از ۳۸ شرکت دریافت شده است. نشست شورای راهبری جویشگر بومی با حضور رییس و اعضای این شورا در محل سازمان فناوری اطلاعات با دستور «ارایه نتیجه نهایی فراخوان، بررسی دستورالعمل گرانت پژوهشی، بررسی نهایی نظام‌نامه مدیریت طرح و فراخوان مشاور خارجی» برگزار شد.

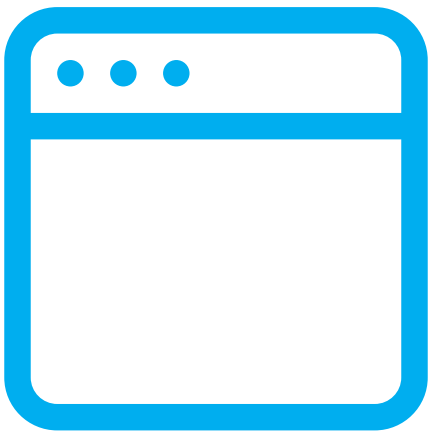
دکتر علیرضا یاری مجری طرح جویشگر بومی معیارهای کلان ارزیابی فنی پیشنهادها را مشخصات محصول (آماده، با کمی تغییر آماده، در حال توسعه، دمو و پژوهش)، توسعه داخلی بودن یا استفاده از محصول‌های متن باز موجود، سابقه ارایه محصول، نوع خدمت‌دهی محصول در حال حاضر (برخط عمومی و خصوصی، آف‌لاین)، شمار کاربران برخط محصول، یکپارچه با دیگر خدمات شرکت، امکانات ویژه در مقایسه با رقبا، معماری مبتنی بر ابر و سرویس پایه برای خدمات طرح بیان کرد.

همچنین درباره پیشنهادهای رسیده برای فراخوان، مقرر شد پیشنهادهای پژوهشی از غیر پژوهشی جدا شده و دانشگاه‌ها با تاسیس شرکت در فراخوان طرح جویشگر بومی شرکت کنند ضمن آن‌که اولویت نخست با سرمایه‌گذاری مشترک خواهد بود.

http://slmd.ir/kz

فایرفاکس صدا را قطع می‌کند

موزیلا در حال کار بر روی ویژگی است تا به وسیله آن بتواند زبانه‌هایی از فایرفاکس که در آن‌ها موسیقی در حال پخش است را مشخص و با نمایش دادن یک آیکون به کاربر بر روی زبانه، امکان قطع صدا را فراهم کند. البته باید اشاره کرد که کروم نیز بیش از این امکان مشخص کردن زبانه در حال پخش را داشت، اما قطع کردن صدا در آن خیلی ساده نبود.شده‌اند. اگر به دنبال دیدن این قابلیت در کروم هستید باید با رفتن به آدرس chrome://flags و کلیک کردن بر روی #enable-tab-audio آن را فعال کنید.



معرفی

معرفی پروژه ژاکارد

انقـلاب دنیای منسوجات و فـنـاوری‌های پوشـیدنی در راه است

مترجم:
نیما صحرانشین سامانی

گـروه فن‌آوری و پروژه‌های پیش‌رفته گوگل در حال تحقیق و توسـعه محصول جدیدی هستند که می‌تواند انقلابی جدید را در عصر فن‌آوری رقـم بزند. این فعالیت در قالب پـروژه ژاکارد در حال پیگیری است. هدف پروژه ژاکارد (Project Jacquard)، ممکن کردن بافت لمسی و تعاملی در هر منسوجی با اسـتفاده از دسـتگاه‌های بافندگی صنعتی اسـت. ژاکارد به طراحان و توسعه‌دهندگان اجازه می‌دهد تا منسوجات حساس به لمس را در محصولات خود به کار ببرند و اشیای روزمره مانند لباس‌ها و مبلمان را به سطوح تعاملی تبدیل کنند.

ریسندگی نخ رسانا

این اتفاق به لطف نخ رسـانای جدید که در همکاری با شریک صنعتی‌مان ساخته شده است، امکان پذیر شده است.

سـاختار نـخ ژاکارد، آلیاژهای فلزی نـازک را با الیاف طبیعی و مصنوعی مانند پنبه، پلی‌اسـتر یا ابریشـم ترکیـب می‌کنـد. در نهایت نـخ به انـدازه کافی قوی می‌شود تا بر روی هر دستگاه صنعتی‌ای بافته شود. نـخ ژاکارد از نخ‌هـای معمولی که امـروزه برای تولید پارچه استفاده می‌شود، غیرقابل تشخیص است.

بافتن پارچه‌های تعاملی

با اسـتفاده از نخ رسانا می‌توان نقاط لمس یا مناطق حساس به حرکت را در محل‌های دقیق در هر نقطه از منسوج بافت.

راه دیگـر، بافتن شـبکه‌ای از حسـگرها در سرتاسـر منسوج است تا سطحی بزرگ و تعاملی ایجاد شود.

لوازم الکترونیکی تعبیه شده

اجزای مکمل کاملا مهندسی شده‌اند تا بتوانند تا حد ممکن گسسته باشند.

ما تکنیک‌های نوآورانه‌ای را توسعه داده‌ایم تا نخ‌های رسانا را به اتصالات و مدارهای کوچکی (کوچک‌تر از دکمه‌های یک کت) متصل کنیم.

این ابزارهای الکترونیکی مینیاتوری، فعل و انفعالات لمسی و حرکات مختلف را ضبط می‌کنند. با استفاده از این داده‌های ضبط شـده و الگوریتم‌های یادگیری ماشـین، امکان استنباط بسیاری از حرکات به وجود می‌آید.

داده‌هـای ضبـط شـده از حرکت و لمس، به شـکل بی‌سـیم به تلفن همراه یا وسـایل دیگر انتقـال داده می‌شـود تا طیف گسترده‌ای از توابع، اتصال کاربر به خدمات ابـری، برنامه‌ها یا قابلیت‌های تلفن همراه را کنترل کند.

ال‌ای‌دی‌هـا، هپتیک‌ها (خروجی را به شـکل لرزش یا فشـار تولید می‌کنند) و خروجی‌های تعبیه شده دیگر، بازخورد را برای کاربر ایجاد می‌کنند و به شکل یک‌پارچه آن‌ها را به دنیای دیجیتال متصل می‌کنند.

تولید در مقیاس

تولید اجزای ژاکارد مقرون به صرفه اسـت و نخ‌ها و پارچه‌ها را می‌توان با استفاده از تجهیزات استانداردی که در سراسر جهان استفاده می‌شود بافت.

یک دستگاه بافندگی می‌تواند طرح‌های مختلفی را به ازای هر فردی در جهان تولید کند

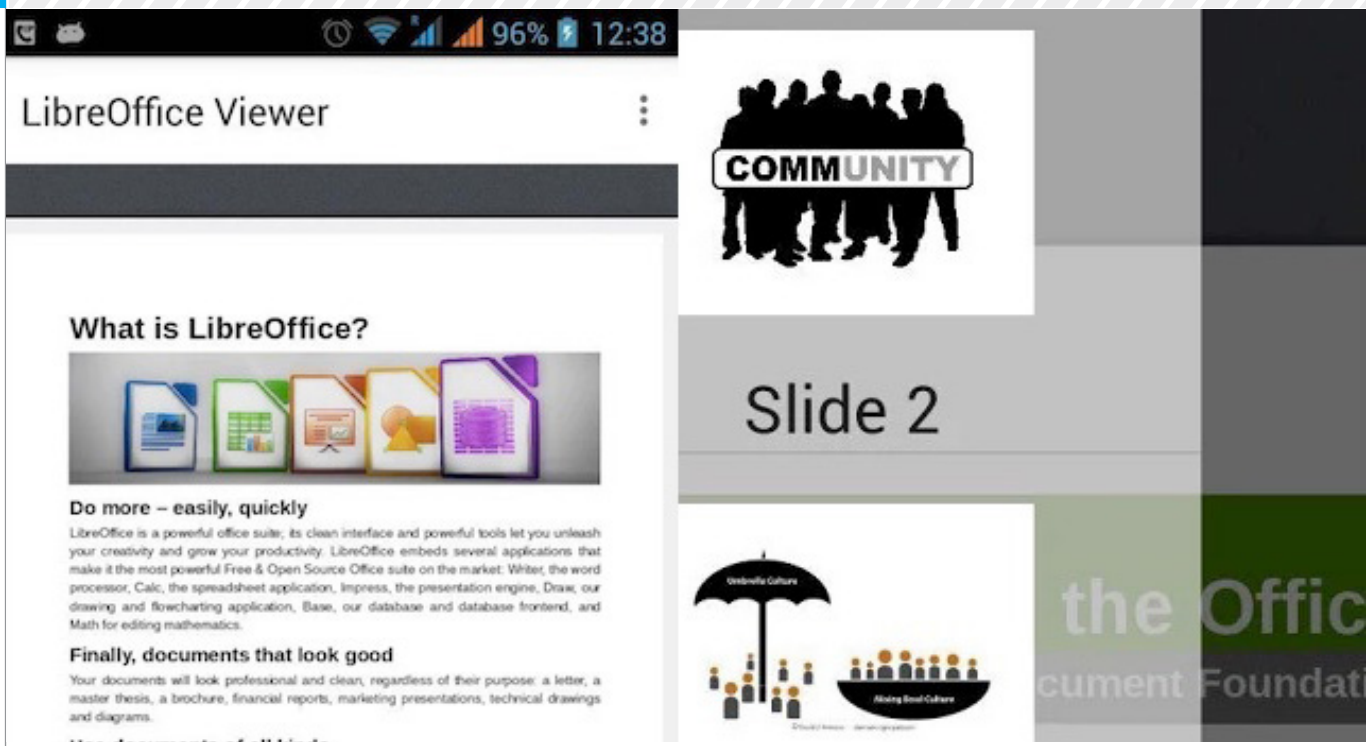
ساخت پوشاک متصل

پوشـاک متصل امکانـات جدیدی را بـرای تعامل با خدمات، وسایل و محیط‌ها می‌دهند. این تعاملات را در هر لحظه‌ای می‌توان دوباره پیکربندی کرد.

ژاکارد بوم سـفیدی برای صنعت مد اسـت. طراحان بدون آن که نیاز به دانستن الکترونیک داشته باشند می‌توانند از آن به جای هر پارچه‌ای استفاده کنند و لایه جدیدی از قابلیت‌ها را به طراحی خود بیفزایند.

توسعه‌دهندگان می‌توانند هر برنامه و خدمت موجود را به پوشاک دارای فن‌آوری ژاکارد خود متصل کنند. و ویژگی جدیدی را برای بن‌سازه خود ایجاد کنند.

هم‌چنیـن متولیان اصلی، در حال توسـعه اتصالات سفارشی، ابزارهای الکترونیکی، پروتکل‌های ارتباطی و یک زیست‌بوم از نرم‌افزارهای ساده و خدمات ابری هستند■



لیبره‌آفیس برای اندروید

قدرت و آزادی را در اندروید خود احساس کنید

لیبره‌آفیس یکی از نرم‌افزارهای آزاد معروف نزد کاربران گنو/لینوکس است. این برنامه از نرم‌افزار آزاد اوپن‌آفیس انشعاب یافته که توسط اوراکل خریداری شد و پس از آن به دلایلی نامعلوم به بنیاد آپاچی منتقل و پروانهٔ انتشار آن نیز به پروانهٔ آپاچی تغییر یافت. لیبره‌آفیس زمانی از اوپن‌آفیس منشعب شد که شرکت سان توسط شرکت اوراکل خریداری شده بود. پس از این انشعاب، بنیادی به نام بنیاد مستندات ایجاد شد که از کارمندان و توسعه‌دهندگان سابق شرکت سان در بخش اوپن‌آفیس بودند.

بنیاد مستندات به تازگی لیبره‌آفیس جدید را برای اندروید نیز عرضه کرد. این برنامه به کاربران اندروید اجازه می‌دهد اسنادشان را خوانده و ویرایش کنند. پیش از این نیز نگارش‌هایی غیررسمی از اوپن‌آفیس یا حتی لیبره‌آفیس برای اندروید منتشرشده بود و در نگارش‌های نخستین برنامه‌ی رسمی نیز، تنها امکان مشاهده‌ی اسناد وجود داشت.

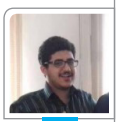
نکته‌ای که بیش‌از همه این محصول را از رقبای خود متمایز می‌کند این است که لیبره‌افیس روی اندروید از همان موتور پردازشی استفاده می‌کند که روی نرم‌افزار اصلی وجود دارد و برای مثال می‌توان آن را نخستین مجموعه‌ای دانست که می‌تواند فرمول‌های ریاضی را در اسناد به درستی نشان دهد. با این حال، این نگارش هنوز یک مجموعه‌ی نرم‌افزارهای اداری کامل روی تلفن‌های هوشمند نیست و از طرف این بنیاد به عنوان یک نرم‌افزار مشاهده‌ی اسناد با امکانات ابتدایی ویرایش، مانند تغییر واژگان، تغییر رنگ و نوع قلم و معرفی شده است.

مشاهده‌ی اسناد در این نگارش از لیبره‌آفیس اندروید احساس آزمایشی بودن را به کاربر منتقل می‌کند و به نظر می‌رسد که هنوز جای کار زیادی دارد. مثلا به‌طور کامل نمی‌تواند بین اسناد شما در حافظهٔ داخلی و خارجی تفاوتی قائل شود و یا با خدمات ابری هماهنگ شده و به اسناد داخل آن‌ها دسترسی داشته باشد؛ هم‌چنین قادر نیست مانند گوگل درایو پشتیبانی از اسناد را در فضای ابری مختص به خود یا ابزاری ثالث، نگه دارد.

نکته‌ای که بیش‌از همه این محصول را از رقبای خود متمایز می‌کند این است که لیبره‌افیس روی اندروید از همان موتور پردازشی استفاده می‌کند که روی نرم‌افزار اصلی وجود دارد

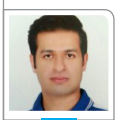
یافتن پرونده‌ها در این نرم‌افزار نیاز به پیمایش در ساختار سامانه پرونده‌ی کم‌تر شهودی اندروید دارد و ممکن است خیلی از افراد پس از برخورد با چنین تجربه‌ای، عطای این برنامه را به لقایش ببخشند. ولی کسانی که می‌مانند، یک مشاهده‌گر اسناد بسیار باکیفیت را می‌یابند، به همراه ویرایشگری که گرچه به اندازه‌ی کافی پایدار است، ولی برای استفاده روی نمایشگر کوچک تلفن خیلی مناسب‌نیست.

افراد زیادی هستند که از ارائهٔ چنین ابزارهایی خوشحال شوند، از جمله کسانی که علاقه دارند از ابزارهای آزاد بیش‌تری در تلفن همراه خود استفاده کنند. دیگر افراد هم که نگاه جدی‌تری به مقوله‌ی مجموعه‌های اداری دارند، باید به این نرم‌افزار نگاه ویژه‌ای داشته باشند.



مترجم
احسان ترک

مکالمه صوتی از طریق مامبل



نویسنده: مهدی حمیدی

مامبل علاوه بر انتقال صدا، قابلیت گپ زدن از طریق تگ‌های HTML و انتقال عکس را نیز فراهم می‌کند. علاوه بر این، مامبل یک API مخصوص دارد که قابلیتی نظیر Mumble Overlay را ارائه می‌کند. این قابلیت که قابل ترکیب با DirectX و OpenGL است باعث می‌شود که در یک بازی مانند دنیای وارکرفت یا دوتا۲، صدای مکالمه‌ی دوست خود را از همان جهتی بشنوید که شخصیت او در بازی قرار دارد. قابلیت استفاده موثر از شبکه تور، در کنار رمزنگاری داده‌های مربوط به صدا، امنیتی مثال‌زدنی به وجود می‌آورد. هم‌چنین مامبل سه حالت استفاده را پشتیبانی می‌کند: حالت فعال شدن دستی، حالت فعال شدن با اوج گرفتن صدا و حالت فعالیت دائم.

توضیح: مامبل از دو بخش تشکیل شده، یک بخش سرور است که در مخازن توزیع‌های بر پایه دبیان به طور معمول با نام بسته mumble-server شناخته می‌شود و بخش کلاینت با همان نام بسته mumble شناخته می‌شود. در ادامه به نصب و پیکربندی mumble-server می‌پردازیم. mumble-server که نام دیگر آن Murmur است، قابلیت دسترسی شخصی یا عمومی را به کاربران می‌دهد.

نصب mumble-server:

مقدار حافظه ۵۱۲ مگابایت برای راه‌اندازی نسخه سرور کافی است، آن‌قدر که ۵۰ مکالمه همزمان بتوانند با کیفیت عالی صورت گیرند. همان طور که اشاره شد، مامبل یک ابزار مستقل از بستر است. به لطف توقع سخت‌افزاری پایین، من توانستم کارساز را روی برد رزبری‌پای به راحتی نصب و استفاده کنم.

نصب و پیکربندی mumble-server در یک توزیع مبتنی بر دبیان گنو/لینوکس:

دستور زیر را در پوسته گنو/لینوکس خود وارد کنید:

```
$ sudo apt-get install mumble-server
```

تبریک! شما به سادگی یک نسخه سرور از Mumble را نصب کردید!

پس از نصب، نیاز است پیکربندی‌های نخستین را انجام دهیم. پس دستور زیر را وارد می‌کنیم:

```
$ sudo dpkg-reconfigure mumble-server
```

در بخش نخست از کاربر پرسیده می‌شود که می‌خواهد Daemon(دیمِن) مربوط به کارساز مامبل در هر بار راه‌اندازی اجرا شود که می‌پذیریم. (تصویر ۱)

در بخش دوم، درباره اولویت اجرای فرایند دیمِن مربوط به مامبل از کاربر پرسیده می‌شود. هر چه اولویت این فرآیند بالاتر باشد، سیستم‌عامل بازه‌های زمانی بیش‌تری را به آن تخصیص می‌دهد که امکان ایجاد وقفه بین مکالمات پایین‌تر می‌رود. (تصویر ۲)

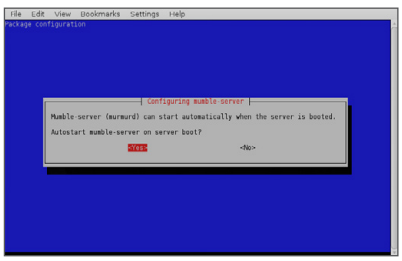
و در نهایت در بخش سوم، گذرواژه کاربر ارشد پرسیده می‌شود. کاربر ارشد قادر است با استفاده از نام کاربری و این گذرواژه وارد تنظیمات کارساز مامبل شده و آن را تنظیم کند. (تصویر ۳)

تنظیمات پیشرفته کارساز مامبل در پرونده /etc/mumble-server.ini قرار دارد. از آن تنظیمات می‌توان به پورت در حال شنود (که به طور پیش‌گزیده مقدار ۶۴۷۳۸ را دارد)، پیغام خوش‌آمد گویی، پهنای باند، سیاست جلوگیری از حملات Brute Force، تعداد بیشینه کاربران هم‌زمان، بیشینه طول پیام گپ، اندازه تصاویر ارسالی و مخفی شدن در شبکه اشاره نمود.

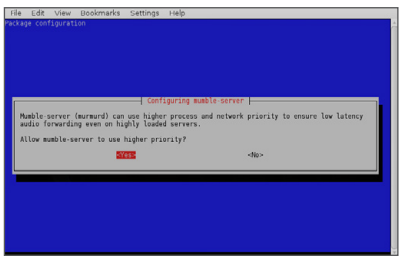
نصب کلاینت‌ها:

کلاینت‌های مختلفی برای اتصال به یک کارساز مامبل نوشته شده‌اند. علاوه بر کلاینت‌های مربوط به رایانه‌های رومیزی، کلاینت‌های مامبل برای اکثر بن‌سازه‌های قابل حمل نیز نوشته شده است. در توزیع‌هایی مانند دبیان و اوبونتو، نصب این کلاینت‌ها به سادگی اجرای دستور زیر است:

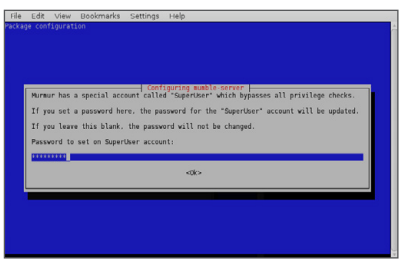
```
$ sudo apt-get install mumble
```



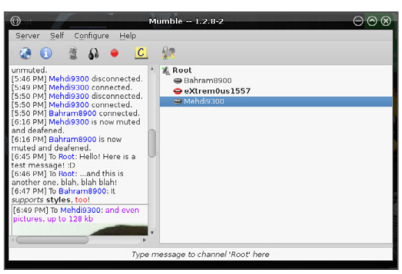
(تصویر ۱)



(تصویر ۲)



(تصویر ۳)



(تصویری از محیط مامبل)

نرم‌افزار آزاد، به داد پرونده‌های پاک شده می‌رسد

تست‌دیسک، ابزار بازگردانی فایل‌های پاک شده

اکثر افرادی که سر و کارشان با رایانه است، به نحوی خاطره‌ای از پاک شدن ناخواسته پرونده‌های مورد نیازشان را دارند. در این میان بعضی از این افراد سعی در بازیابی داده‌های پاک شده (اصطلاحا ریکاوری) نموده‌اند که با مشکلاتی مانند عدم بازیابی کامل و.. . روبرو شده‌اند. ولی کم‌تر جایی سخن از نرم‌افزار آزاد تست‌دیسک که ابزاری قدرت‌مند و قابل اجرا بر روی سیستم عامل‌های متعددی است برده شده. در این راهنما قصد داریم شما را به صورت عملی با سناریوی بازگردانی داده‌های پاک شده از روی حافظه‌های جانبی مانند دیسک سخت، فلش، کارت‌های حافظه و آشنا کنیم.

نویسنده:
بهنام توکلی

معرفی تست‌دیسک:

تست‌دیسک (به انگلیسی TestDisk) یک نرم‌افزار آزاد برای بازگردانی داده‌های پاک شده است. این برنامه عمدتا برای بازگردانی پارتیشن‌های مفقود شده است و یا برای قابل راه‌اندازی (بوتیبل) کردن دیسک‌هایی که این گونه نیستند طراحی شده. این اشکالات می‌توانند به دلایل مختلفی از جمله خطاهای نرم‌افزاری، ویروس‌ها و یا خطاهای انسانی (همانند پاک کردن جدول پارتیشن به صورت تصادفی) به وجود آیند.

با استفاده از تست‌دیسک، می‌توان اطلاعات دقیق و مشروحی را در مورد دیسک‌های خراب به دست آورد و سپس آن اطلاعات را به منظور تحلیل و بررسی بیشتر برای یک متخصص ارسال کرد. تست‌دیسک روی سیستم‌عامل‌های مختلفی از جمله داس، ویندوز، مک اواس ده، گنو/لینوکس، فری‌بی‌اس‌دی، اوپن‌بی‌اس‌دی، نت‌بی‌اس‌دی و اجرا می‌شود. تست‌دیسک هم‌چنین از بسیاری از طرح‌بندی‌های رایج جدول پارتیشن پشتیبانی می‌کند. از جمله طرح‌بندی‌های ام‌بی‌آر، جی‌پی‌تی و تعدادی دیگر. تست‌دیسک می‌تواند پارتیشن‌های حذف شده را بازیابی کند، جدول پارتیشن را بازسازی کند و رکورد راه‌اندازی اصلی درایو را بازنویسی کند. وقتی که پرونده‌ای از روی دیسک حذف می‌شود، فهرست خوشه‌هایی که پرونده اشغال کرده بود هم پاک شده و پرونده‌های دیگر می‌توانند از این سکتورها برای ذخیره اطلاعات خود استفاده کنند. اگر پرونده تکه‌تکه نشده باشد و خوشه‌ها توسط پرونده‌های دیگر استفاده نشده باشند، تست‌دیسک قادر است پرونده مورد نظر را بازیابی کند. این برنامه از فایل‌سیستم‌های متعددی پشتیبانی می‌کند و تقریبا می‌توان گفت تست‌دیسک از همه فایل‌سیستم‌های رایج در سیستم‌عامل‌های مختلف پشتیبانی می‌کند. در جولای ۲۰۰۸، تست‌دیسک و فوتورک (در قالب یک برنامه) بیش از ۱۵۰۰۰۰ بار فقط از پایگاه وب اصلی دریافت شدند.

این برنامه‌ها به قدری محبوب هستند که برخی از دیسک‌های زنده گنو/لینوکس از جمله ناپیکس، جی‌پارتد، پارتد مجیک و به صورت پیش‌گزیده به همراه تست‌دیسک عرضه می‌شوند. [نقل شده از ویکی‌پدیا]

گام به گام بازگردانی پرونده(های) پاک شده از حافظه:

پیش از شروع به انجام هر کاری، باید یادآور شویم که تست‌دیسک دربردارنده قابلیت‌های بسیاری است که بازگردانی پرونده‌های پاک شده یکی از آن‌ها است. تست‌دیسک با توجه به نوع قابلیت، از فایل‌سیستم‌های مختلفی پشتیبانی می‌کند که فایل‌سیستم‌های پشتیبانی شده جهت قابلیت بازگردانی پرونده‌های پاک شده شامل FAT، NTFS، exFAT و خانواده Ext است. بنابراین خیال ویندوزی‌ها از بابت پشتیبانی از کلیه فایل‌سیستم‌های حافظه‌های جانبی‌شان راحت باشد، ولی کاربران دیگر سیستم‌عامل‌ها حتما پیش از استفاده از این قابلیت به نوع فایل‌سیستم حافظه جانبی مورد نظر دقت داشته باشند.

۱. بارگیری / نصب:

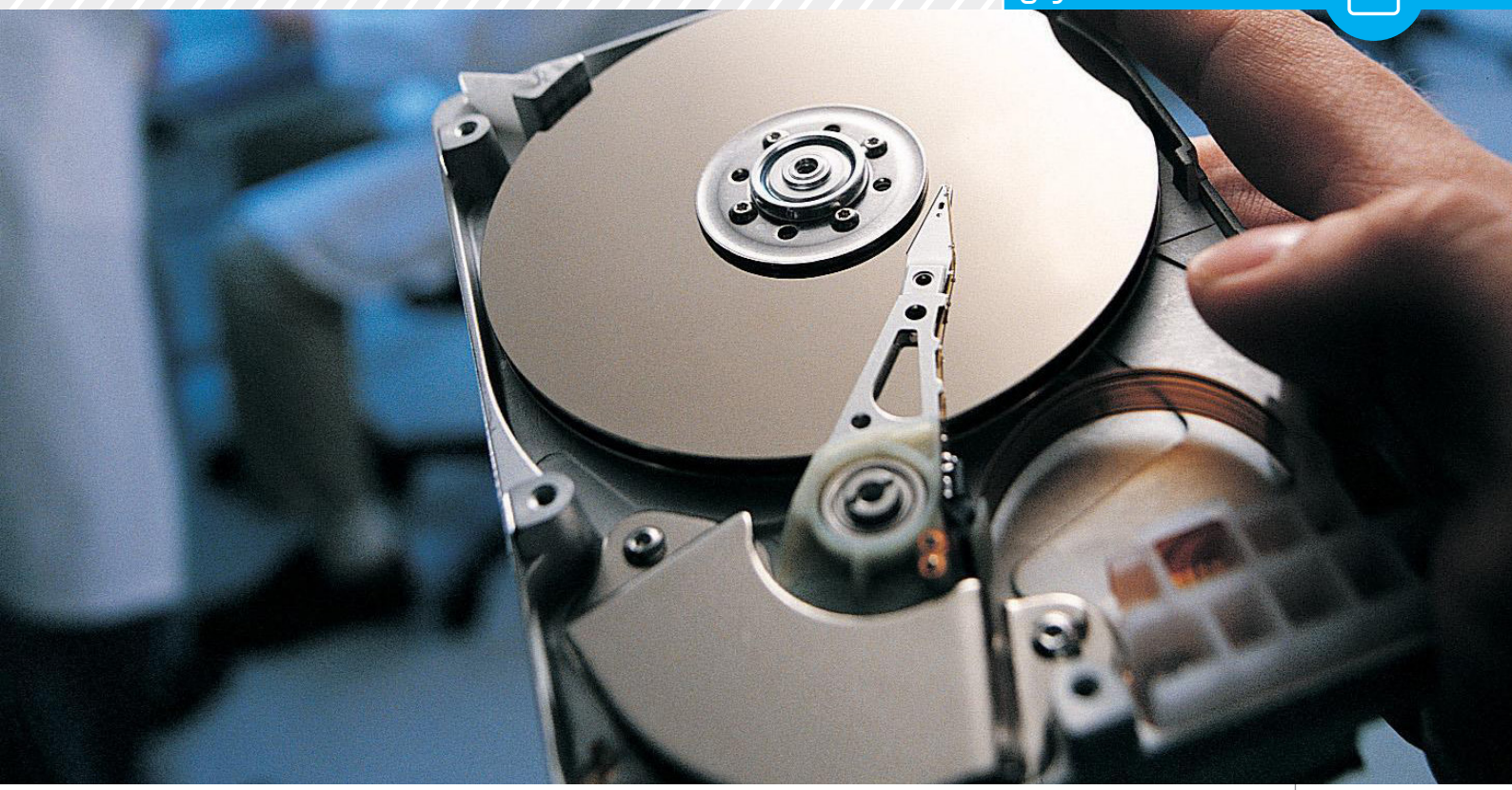
ابتدا با مراجعه به صفحه دریافت تست‌دیسک پرونده اجرایی برنامه را دریافت کنید. خوشبختانه تست‌دیسک دارای نگارش‌های ویندوز، گنو/لینوکس، مک، داس و است.

اگر کاربر اوبونتو یا دبیان گنو/لینوکس هستید می‌توانید با استفاده از رابط گرافیکی مدیریت بسته‌های نرم‌افزاری یا اجرای دستور زیر در خط فرمان اقدام به نصب این نرم‌افزار کنید:

```
sudo apt-get update; sudo apt-get install testdisk
```

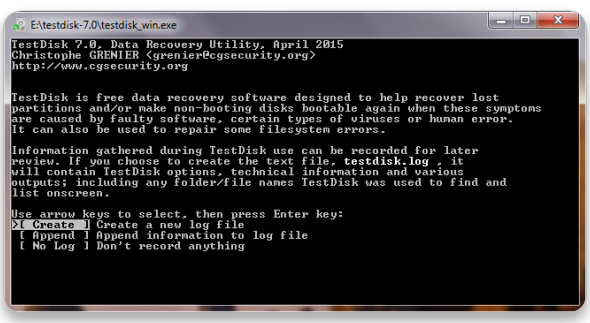
هم‌چنین اگر کاربر سنت‌اواس یا دیگر توزیع‌های گنو/لینوکس مبتنی بر فدورا هستید، می‌توانید با استفاده از رابط گرافیکی آن توزیع گنو/لینوکس یا اجرای دستور زیر در خط فرمان اقدام به نصب این نرم‌افزار کنید:

```
su; yum update; yum install testdisk
```

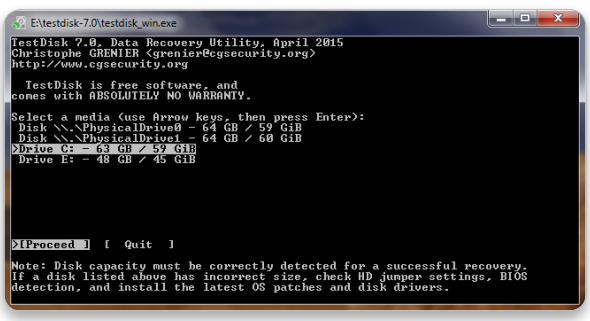


اجرا و بازگردانی پرونده(ها):

اگر از نگارش ویندوزی استفاده می‌کنید، باید پرونده تست‌دیسک را از حالت فشرده خارج کرده و پرونده testdisk_win.exe را اجرا کنید. اگر کاربر گنو/لینوکس هستید با زدن دستور testdisk در محیط خط فرمان به عنوان کاربر ریشه می‌توانید وارد محیط برنامه شوید (جهت اجرا به صورت کاربر ریشه در توزیع اوبونتو کافی است از دستور sudo testdisk استفاده کنید). توجه داشته باشید که testdisk دارای رابط گرافیکی نیست، ولی کار با رابط متنی آن بسیار ساده و درهمه سیستم‌عامل‌ها به یک نحو است. پنجره ای مطابق با تصویر ۱.



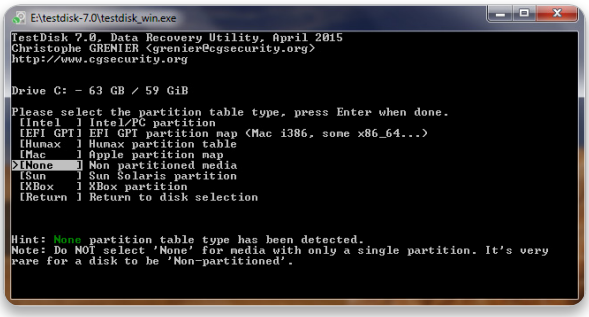
۱ پس از اجرای برنامه با پنجره روبه‌رو مواجه می‌شوید. اگر تمایل دارید که کلیه فعالیت‌هایی که انجام می‌شود در یک پرونده متنی ثبت شود باید گزینه نخست را جهت ایجاد یک پرونده جدید یا گزینه دوم را جهت اضافه شدن گزارش های جدید به پرونده‌ای که از پیش ایجاد شده است انتخاب کنید. اما اگر نیازی به این گزارش ندارید یا فایل‌سیستم حافظه جانبی شما به صورت فقط خواندنی درآمده است، باید گزینه آخر یعنی No Log را برگزینید.



۲ در مرحله بعدی از شما حافظه جانبی‌ای که قصد انجام فعالیت روی آن را دارید، پرسیده می‌شود و باید با کلیدهای بالا/پایین آن را برگزینید.

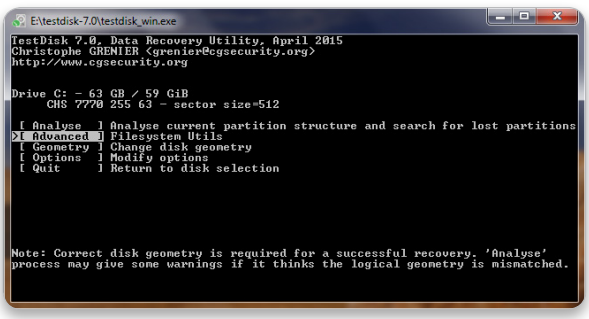
۳

در مرحله بعد، تست‌دیسک نوع جدول پارتیشن حافظه جانبی را می‌پرسد که باید گزینهٔ مناسب را انتخاب کنید. در این‌جا گزینه None که پیش‌گزیده است را برمی‌گزینیم.



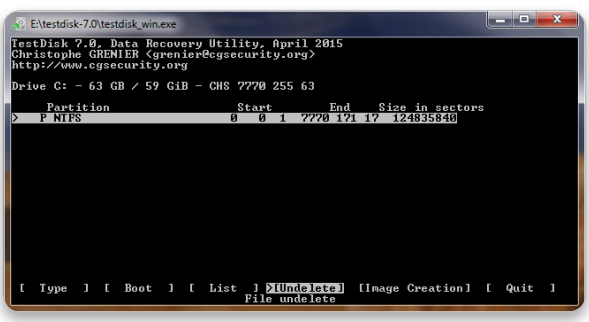
۴

در مرحله بعدی باید قابلیت مورد نظر جهت انجام روی حافظه انتخاب شده را انتخاب کرد که جهت بازگردانی پرونده‌های پاک شده باید گزینه Advanced را برگزید.



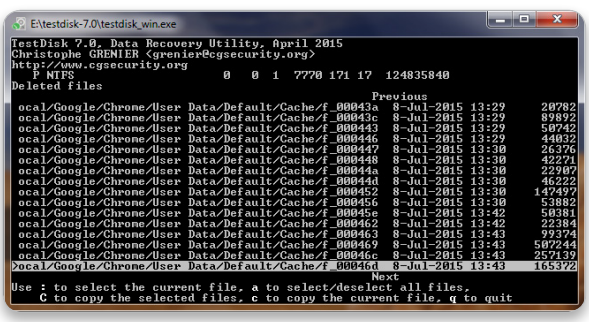
۵

سپس فهرستی از پارتیشن‌های حافظه جانبی انتخاب شده نمایش داده می‌شود که باید پارتیشن مورد نظر را با استفاده از کلیدهای بالا/پایین انتخاب کرده و سپس با کلیدهای چپ/راست گزینه Undelete را از گزینه‌های عملیاتی پایین برگزیده و اجرا کنید.



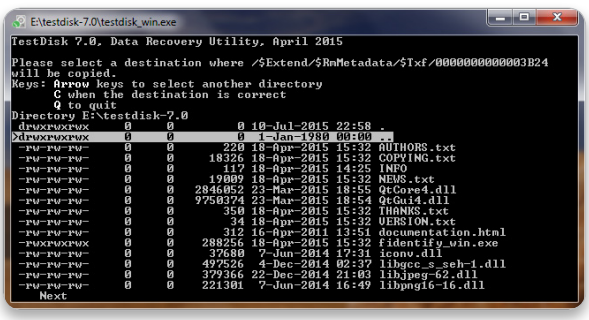
۶

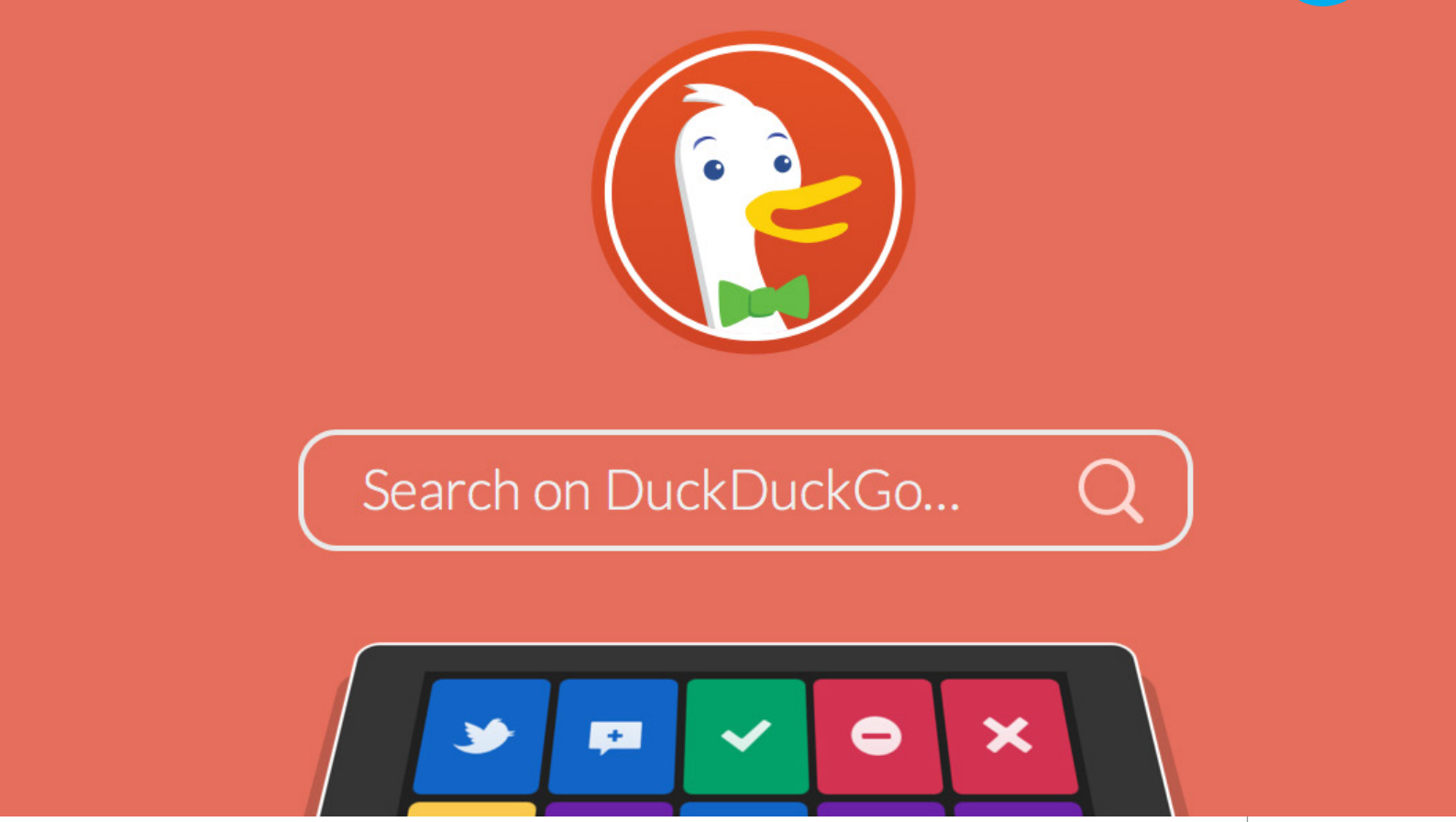
سپس فهرستی از پرونده‌های پاک شده روی پارتیشن مورد نظر نمایش داده می‌شود که با استفاده از دکمه‌های بالا/پایین می‌توانید ادامه فهرست را مشاهده کنید. جهت انتخاب یک پرونده خاص، روی آن پرونده رفته و علامت «:» را بنویسید. هم‌چنین برای انتخاب کلیه پرونده‌های نمایش داده شده می‌توانید حرف «a» را بنویسید. موارد انتخاب شده به صورت رنگی نمایش داده خواهند شد.



۷

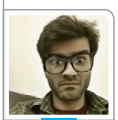
سپس برای رونوشت گرفتن از پرونده‌ای که روی آن قرار دارید، می‌توانید حرف C (سی بزرگ) را نوشته یا برای رونوشت گرفتن از پرونده‌های انتخاب شده (توسط علامت : یا نوشتن حرف a) می‌توانید حرف c (سی کوچک) را بنویسید.در مرحله بعدی باید مشخص کنید که پرونده‌های انتخاب و رونوشت شده در کجا قرار داده شوند. در این مرحله باید مسیر مورد نظر را مشخص کنید. لازم به ذکر است که عبارت «.» (یک نقطه) نمایانگر مسیر فعلی و عبارت «..» (دو نقطه کنار هم) نمایانگر شاخه والد فولدر(پیشین) است. سرانجام هنگامی که به مسیر مورد نظر رفتید با نوشتن حرف C (سی بزرگ) پرونده‌های انتخاب شده شروع به رونوشت شدن در آن مسیر می‌کنند و بدین صورت کار بازیابی پرونده‌های پاک شده به پایان می‌رسد.■





با خیال راحت در وب جستجو کنید

۷ برتری جست‌وجوگر داک‌داک‌گو نسبت به گوگل و بینگ



نویسنده:
نیوشا شیر

داک‌داک‌گو (duckduckgo) یک موتور جست‌وجوی وب است که در عملکرد خود تا حد زیادی بر داده‌های جمع‌سپاری‌شده هم‌چون ویکی‌پدیا تکیه دارد. یکی از نکات مهم این موتور جست‌وجو (به ادعای خودش)، عدم ردگیری کاربرانش است. این موتور جست‌وجوی وب را می‌توان حاصل نگرانی‌ها پیرامون حریم خصوصی افراد دانست. در صفحه سیاست محرمانگی این پایگاه وب آمده است که هیچ گونه اطلاعات شخصی‌ای را جمع‌آوری نکرده و به اشتراک نمی‌گذارد. بر خلاف دیگر موتورهای جست‌وجو در سیاست داک‌داک‌گو آمده است که تاریخچه جست‌وجوهای کاربران را ذخیره نمی‌کند.

از ویژگی‌های جست‌وجو در این موتور، می‌توان به شیوه جست‌وجوی واژه‌های دارای ابهام در آن اشاره کرد. بر خلاف برخی از شیوه‌های دیگر جست‌وجو که بر پایه بهترین نتایج در صفحه نخست کار می‌کنند، داک‌داک‌گو تلاش می‌کند تمام معانی مختلف یک واژه تفسیرپذیر را در صفحه نخست بگنجاند تا نتایج این صفحه به یک معنی خاص سوگیری نداشته باشد. افزون بر این، کاربر را راهنمایی می‌کند تا مقصود خود را بهتر بیان کند.

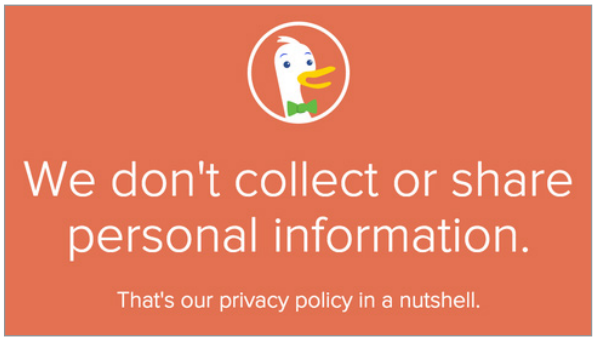
بنابراین، داک‌داک‌گو بسیار جذاب است. در بسیاری از موارد، همانند موتورهای جست‌وجوی دیگری مثل گوگل و بینگ است، ولی ترفندهایی در آستین دارد که آن را از رقبایش جدا می‌کند.

محرمانگی و امنیت

این بزرگ‌ترین نقطه تمایز داک‌داگ‌گو است. برخلاف بقیه موتورهای جست‌وجو معروف، داک‌داک‌گو هیچ گونه اطلاعات شخصی را جمع نمی‌کند.

همین مسأله باعث می‌شود که به شکل پیش‌گزیده، هیچ تاریخچه‌ای از جست‌وجو، نشانی‌های ip و کوکی‌ها را ذخیره نکند. علاوه بر این، برای امنیت بیش‌تر، داک‌داک‌گو به شکل خودکار نگارش‌های رمز شده (HTTPS) را نمایش می‌دهد. البته به همین‌جا محدود نمی‌شود و می‌توانید با تور از داک‌داک‌گو استفاده کنید. برای اطلاعات بیشتر سری به این‌جا بزنید:

https://duckduckgo.com/privacy



بنگ

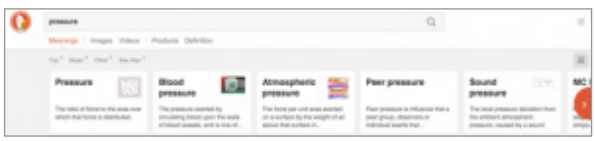
این یکی از ویژگی‌های برجسته شده داک‌داک‌گو است که به شما امکان می‌دهد مستقیما جست‌وجو را روی پایگاه‌های وب دیگر انجام دهید. مثلا من می‌توانم بنویسم «gmail quora!» و داک‌داک‌گو مستقیما برای من «quora» را در حساب gmail من جست‌وجو کند. یا می‌توانم بنویسم «amazon ps4 controller!» و داک‌داک‌گو مستقیما به این صفحه می‌رود: http://slmd.ir/lz

هزاران پایگاه وب وجود دارد که برایشان میان‌برهای بنگ وجود دارد. اگر نتوانستید میان‌بر بنگ را برای پایگاه وبی پیدا کنید، خب بسازیدش. آن‌طور که داک‌داک‌گو گفته: «هشدار: استفاده از بنگ روش جست‌وجوی شما را برای همیشه تغییر می‌دهد. وقت آزاد شما را بیشتر می‌کند و امید به زندگی بیش‌تری پیدا می‌کنید.» واقعا این جمله درست است. یک بار که به سراغ بنگ بروید معتادش می‌شوید و دیگر تمام. برای اطلاعات بیشتر http://slmd.ir/mo را ببینید.

صفحات دسته‌بندی

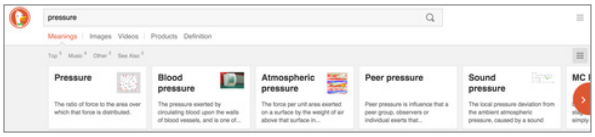
این یکی از ویژگی‌های مورد علاقه من در داک‌داک‌گو است. اساسا می‌توانید صفحات دسته‌بندی را صدا بزنید تا جواب سریعی را در دسته بندی خاصی پیدا کنید. این ویژگی با آماده کردن جواب‌های سریع، متفاوت کوتاه و گزینه‌های معنی‌دار مربوط به عبارتی که جست‌وجو کرده‌اید، در زمانتان صرفه جویی می‌کند. مثلا زمانی که برای «pressure» جست‌وجو کردم، دسته‌بندی‌های زیر را به من نشان داد.

به طور مشابه زمانی که در مورد «south park characters» جست‌وجو کردم، به



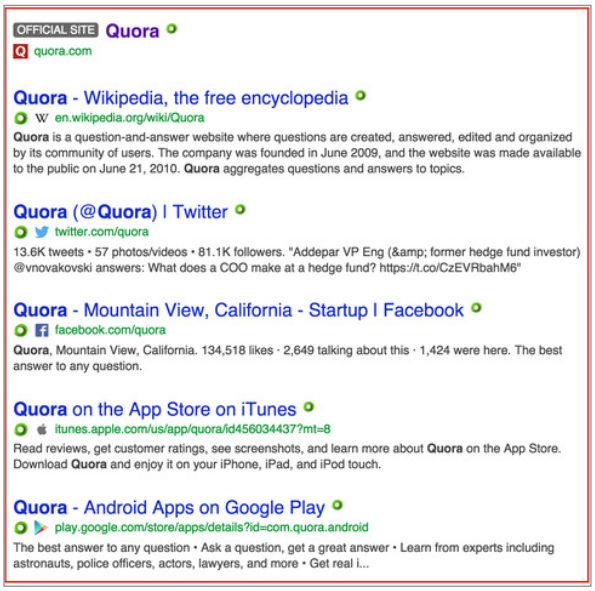
این دسته‌ها رسیدم.

علاوه بر این که پاسخ‌های فوری را ایجاد می‌کند، پرونده ارائه‌ای که پیدا کرده بود نیز خیلی خوب بود.



جست‌وجوی منطقه‌ای

کاربران داک‌داک‌گو می‌توانند نتایج مرتبط با یک منطقه مکانی را با تغییر منطقه به‌دست آورند. به عنوان مثال من «quora» را در حالی که منطقه‌ام ایالات متحده امریکا بود، جست‌وجو کردم و به نتایج زیر رسید.



حالا اگر منطقه را به روسیه تغییر یدهم، نتایج به شکل زیر خواهند شد.

پس با یک کلیک، می‌توان نتایج را برای کاربری با منطقه دیگر ایجاد کرد.

جست‌وجوی بدون تبلیغات

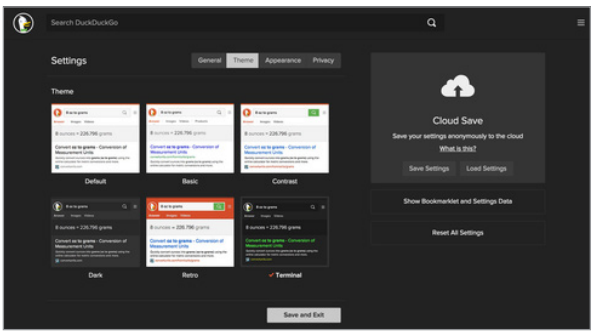
با این‌که به شکل پیش فرض در داک‌داک‌گو تبلیغات نشان داده می‌شود، اما می‌توان آن را با یک کلیک متوقف کرد. برای این کار به مسیر تنظیمات پیشرفته عمومی تبلیغات بروید و خاموشش کنید.

این که بتوانیم به عنوان یک کاربر امکان کنترل داشته باشیم، اتفاق مبارک و خوبی است، در زمانی که هیچ‌کدام از جست‌وجوگرهای گردن کلفت هم این ویژگی را ندارند. البته من باید توضیح بدهم که این تبلیغات بدون جاسوسی، مرتبط و تمیز هستند. به همین دلیل من غیرفعالشان نمی‌کنم، شاید با این کار بتوانم از شرکت مربوط به آن پشتیبانی کنم.

شخصی سازی فراوان

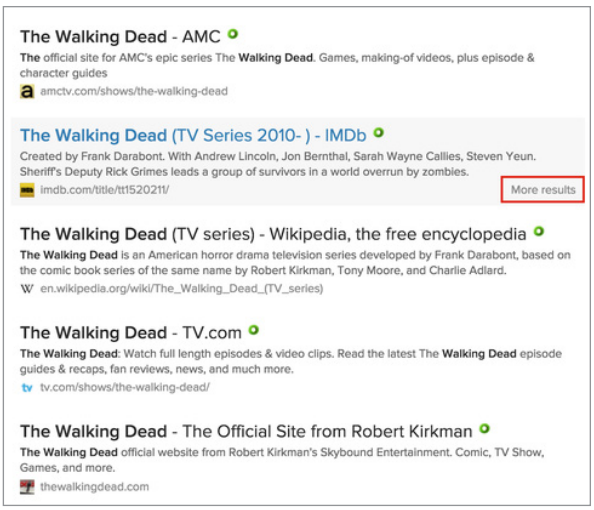
داک‌داک‌گو به کاربرانش امکان شـخصی کردن ظاهر را می‌دهد. مثلا کاربران می‌توانند قالب، قلم‌ها و رنگ‌ها را عوض کنند. تصویر زیر نمایش نتایج با قالب «خط فرمان» است، چیزی شبیه به حالت شب.

عـلاوه بر این، کاربران می‌توانند زبان را تغییر دهند، میان‌برهای صفحه‌کلید را عوض کنند و نشان پایگاه‌های وب در نتایج را ببینید. آخرین و بهترین امکان شخصی‌سازی، امکان ذخیره تنظیمات به صورت ناشناس است.



جست‌وجوی داخل سایت

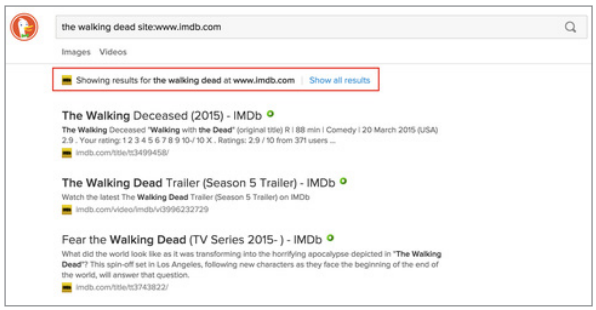
اگرچه گوگل و بینگ هردو امکان جست‌وجو در داخل وب سایت را دارا هستند، ولی پیاده سازی داک‌داک‌گو ساده‌تر و کاربر پسندتر است. علاوه بر این، به ازای هـر نتیجه جسـت‌وجو، گزینه «نتایج بیش‌تر» را نمایـش می‌دهد که با کلیک کـردن رویـش، داک‌داک‌گو تنها نتایج آن پایـگاه وب خاص را نمایش می‌دهد. می‌توانـم روی هـر نتیجه‌ای بروم، آن‌گاه گزینـه «انتخاب‌های بیش‌تر» نمایش



داده می‌شود. در این مثال من روی «IMDb» رفته‌ام.

همان طور که در شـکل مشـخص اسـت، با کلیک کردن روی «نتایج بیش‌تر» فقط نتایج «IMDb» نمایش داده می‌شود.

خیلی ساده شد، این طور نیست؟



ویژگی‌های دیگر

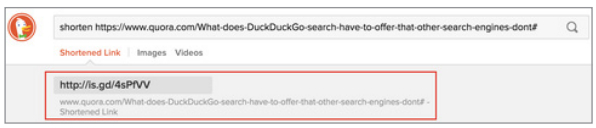
در ادامه برخی ویژگی‌های جذاب و کوچک داک‌داک‌گو را آورده‌ام.

صفحه نتایج قابل لغزیدن:

پیش‌گزیده داک‌داک‌گو این اسـت کـه نتایج را صفحه بندی نکند، پس خیلی ساده می‌توان به پایین لغزید و نتایج بیش‌تری را پیدا کرد. البته اگر نوع صفحه بندی شده را دوست داشته باشید، امکان فعال کردنش وجود دارد.

کوتاه کردن و بسط دادن نسانی‌ها:

می‌توانید با داک‌داک‌گو در یک ثانیه نشـانی هر آدرسـی را کوتاه کرده یا بسط یافته‌اش را پیدا کنید. برای کوتاه کردن تنها کافی است عبارت کلیدی «short-er» را پیش از آدرس بنویسید و داک‌داک‌گو کوتاه‌شده آن را ایجاد می‌کند.



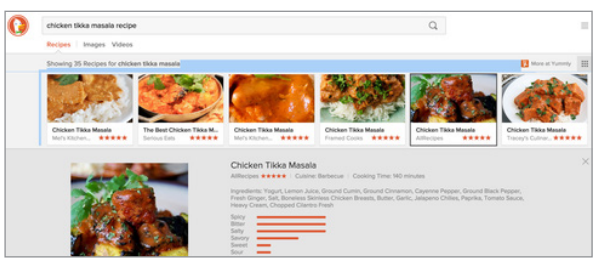
به طور مشابه برای بسط دادن یک پیوند از کلیدواژه «expand» استفاده کنید. لطف این کار در این است که بدون باز کردن پیوند می‌توانید متوجه شوید که نشانی به کجا می‌رود.

حسابگر پرداخت:

داک‌داک‌گو می‌تواند به شما کمک کند تا پرداخت ماهانه وام خود را به آسانی حساب کنید. برای این منظور از کلیدواژه «loan» استفاده کنید. مثل تصویر زیر:

یابنده دستور آشپزی:

می‌توانید دستور پخت غذاهای مختلف را با به کار بردن «recipe» در جست‌وجو، پیدا کنید. داک‌داک‌گو دستورات مختلف را در قالبی زیبا به شما نشان می‌دهد. درست مثل تصویر زیر:

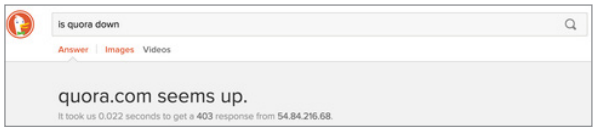


یابنده رنگ:

با داشتن کد مبنای شانزده هر رنگی، می‌توانید آن رنگ را پیدا کنید. برای مثال:

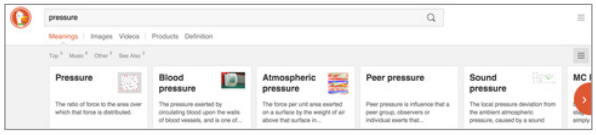
بررسی بالا بودن پایگاه وب:

داک‌داک‌گو در جعبه جست‌وجوی خود به آسانی امکان بررسی کار کردن یک پایگاه وب را می‌دهد.



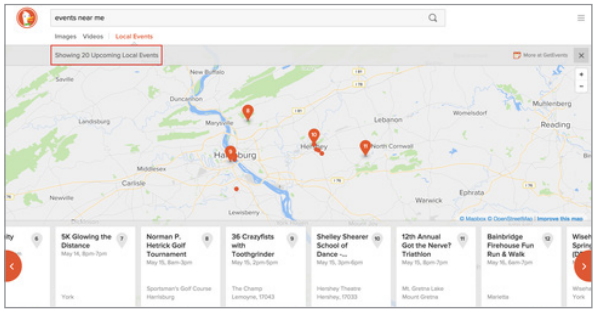
یابنده جایگزین:

می‌توانید جایگزین‌هایی را برای برنامه‌ها، پایگاه‌های وب، نرم‌افزارها یا چیزهای دیگر، با استفاده از داک‌داک‌گو پیدا کنید.



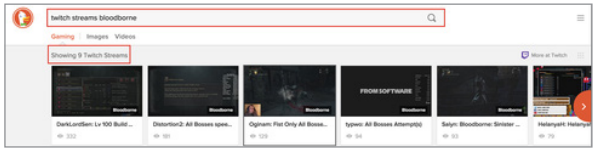
یابنده رخدادها:

با پرسش «event near me» در داک‌داک‌گو می‌توانید رخدادهای آینده موجود در اطراف خود را ببینید.



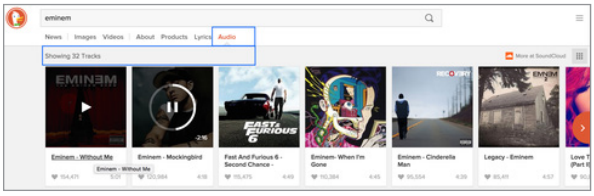
جریان‌های زنده Twitch:

اگر دوست‌دار تماشای جریان‌های زنده بازی ویدیویی بر روی Twitch باشید، داک‌داک‌گو این کار را در صفحه جست‌وجوی خود ساده کرده است.



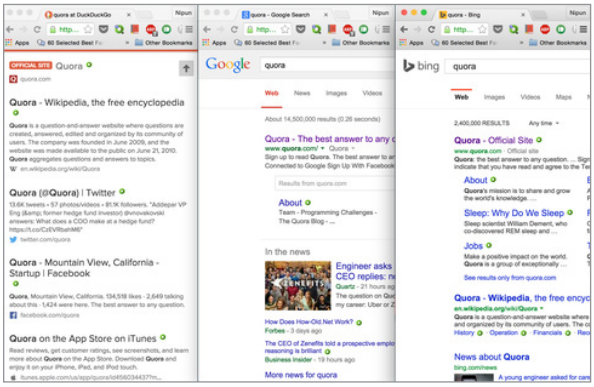
موسیقی:

در داک‌داک‌گو می‌توانید به آهنگ‌های هنرمندان در صفحه نتایج جست‌وجو گوش دهید. مثلا زمانی که «Eminem» را جست‌وجو کنم، می‌توانم بدون ترک کردن صفحه جست‌وجو، معروف‌ترین آهنگ‌هایش را گوش بدهم.



طراحی واکنش‌گرا:

این مورد بیشتر به تجربه کاربری مربوط است تا به ویژگی‌ها، ولی صفحه نتایج داک‌داک‌گو واکنش‌گرا است (خود را با اندازه صفحه تطبیق می‌دهد). در حالی که گوگل و بینگ این‌طور نیستند. به مقایسه زیر دقت کنید.



برای آشنایی با ویژگی‌های جذاب دیگر به صفحه ویژگی‌ها بروید. این صفحه شامل تعداد زیادی از ویژگی‌های کاربردی مثل حل کننده مغلوب، مردم در فضا و کمیک‌های xkcd در داک‌داک‌گو است.

خلاصه این‌که بهترین چیز در مورد داک‌داک‌گو، قدرت دادن به کاربر برای کنترل است. علاوه بر این که خیلی جذاب هم هست. فقط داکش کنید (duck it)!

معرفی ۸ افزونه کاربردی برای مرورگر گوگل کروم

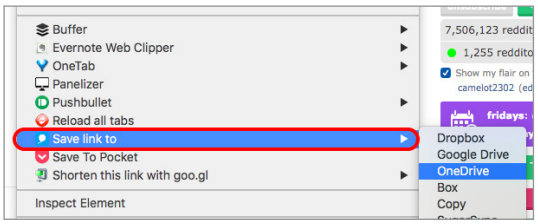
اکثر کاربران مرورگرهای پیش‌رفته امروزی نیاز دارند تا از افزونه‌های آماده و نوشته شده برای مرورگرها استفاده کنند تا برخی تنظیمات و قابلیت‌هایی که به صورت عادی در مرورگرها وجود ندارد را به مرورگر خود بیفزایند. مرورگرهای امروزی مانند مرورگر گوگل کروم، فایرفاکس و اپرا یا حتی سافاری، این روزها از قابلیت افزودن افزونه‌ها پشتیبانی می‌کنند و به راحتی قادر هستید که در این مرورگرها، افزونه‌های خود را نصب کنید. در این بین اگر کاربر گوگل کروم باشید با فهرست بلند بالایی از افزونه‌های قابل نصب در این مرورگر مواجه خواهید شد که به راحتی قادر هستند اکثر امور ساده و حتی پیش‌رفته مورد نیاز شما در وب‌گردی را تأمین کنند. این ابزارها که به افزونه یا ابزار توسعه مرورگر معروف هستند، می‌توانند قابلیت‌هایی را برای مرورگر کروم و یا کرومیوم فراهم کنند که به صورت پیش‌گزیده در این مرورگر قابل انجام نیستند. در این مقاله به بررسی هشت افزونه کاربردی و مفید در مرورگر کروم خواهیم پرداخت که با نصب آن‌ها، وب‌گردی راحتی را تجربه خواهید کرد

با وجود آن که افزودن افزونه جدید به فهرست افزونه‌های موجود در پایگاه اینترنتی مرورگر موزیلا فایرفاکس بسیار کندتر از قرار دادن افزونه در فهرست افزونه‌های گوگل کروم در پایگاه اینترنتی فروشگاه افزونه‌های گوگل کروم است، این مسأله به دلیل بررسی‌های بیش‌تری است که توسط تیم توسعه مرورگر فایرفاکس برای افزونه‌ها رخ می‌دهد. با وجود این، برخی توسعه‌دهندگان ممکن است از این بررسی‌های طولانی مدت راضی نباشند. در هر حال، مرورگر کروم با فهرستی از افزونه‌های بزرگ و کوچک همراه است که دسترسی هر یک از این افزونه‌ها به اطلاعات شما در آن مشخص شده‌است. وقتی قصد نصب یک افزونه را دارید، بهتر است فهرست دسترسی‌های آن افزونه را مطالعه کنید تا اگر مورد اضافی در اختیارات آن وجود داشت، در نصب آن افزونه تجدید نظر کنید. از ویژگی‌ها و برتری‌های مرورگر گوگل کروم نسبت به مرورگر موزیلا فایرفاکس می‌توان به عدم نیاز این مرورگر به راه‌اندازی مجدد مرورگر پس از هربار نصب افزونه‌ها اشاره داشت. با وجود این که برخی از افزونه‌های مرورگر فایرفاکس نیازی به راه‌اندازی مجدد ندارند؛ ولی متأسفانه خیلی از آن‌ها هنوز هم برای فعال شدن پس از نصب، نیاز به راه‌اندازی مجدد مرورگر دارند.

نمایش ویدیو از طریق کروم‌کَست Chrome Cast

نوعی دانگل اچ دی ام آی (به انگلیسی: HDMI) ساخت شرکت گوگل است که در ۲۴ ژوئیه ۲۰۱۳ عرضه شد. قیمت کروم‌کَست گوگل برابر ۳۵ دلار است که در مقایسه با رقبای خود نظیر فایر تی‌وی آمازون، دارای قیمتی بسیار ارزان است از کروم‌کَست برای پخش خدمات برخط شبکه‌هایی چون نت‌فلیکس و سرویس‌های گوگل مانند گوگل‌پلی و یوتیوب و پرونده‌های رسانه‌ای که روی دستگاه‌های اندرویدی قرار دارند استفاده می‌شود. کروم‌کَست را می‌توان یک جایگزین ارزان قیمت برای ابزارهای نمایش ویدیو و چندرسانه‌ای از طریق تلویزیون دانست که رقیب اصلی آن که توسط اپل ایجاد شده است، اپل تی‌وی نام دارد. در مرورگر کروم، افزونه‌ای قرار دارد که با نصب آن در این مرورگر، هنگام مشاهده یک ویدیو یا شنیدن موسیقی از طریق اینترنت، می‌توانید با استفاده از یک دکمه، آن را برای ادامه پخش به کروم‌کَست ارسال کنید. در این هنگام اگر کروم‌کَست شما از طریق اچ‌دی‌ام‌آی به تلویزیون و اینترنت وصل باشد، قادر خواهد بود تا ادامه آهنگ و ویدیو را در تلویزیون به نمایش بگذارد. این افزونه از طریق صفحه افزونه‌های گوگل کروم قابل دریافت و نصب است. گفتنی است این افزونه علاوه بر این که قادر است ویدیو و آهنگ را به کروم‌کَست ارسال کند، قادر است که صفحات بازدید شده شما را نیز با استفاده از کروم‌کَست روی تلویزیون نمایش دهد.

افزونه بالن «Balllon»

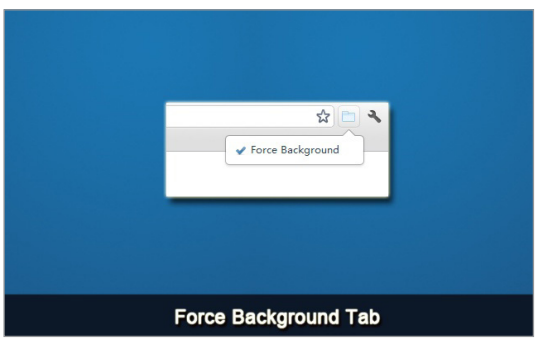


اگر از نحوه نوشتن لاتین کلمه بالن (بادکنک) در نام این افزونه تعجب کرده‌اید و گمان می‌کنید این یک اشتباه تایپی است، در اشتباه هستید. نام این افزونه از سه حرف ال انگلیسی تشکیل شده است. شاید این نام به این دلیل به این شکل نوشته شده‌است که توسعه‌دهندگان از استفاده از نام بالن به شکل صحیح معذوریت قانونی داشته‌اند. در هر حال نام برنامه به این شکل است و اشتباهی در نوشتن آن رخ نداده است! این افزونه برای افرادی مناسب خواهد بود که معمولا اطلاعات زیادی را در فضاهای مختلف ابری بارگذاری می‌کنند. برای نمونه اگر شما اطلاعات خود را در چند خدمت ابری رایگان مانند دراپ‌باکس، وان‌درایو، گوگل درایو و... دارید، با استفاده از این افزونه خواهید توانست با تنها یک کلیک، اطلاعات خود را به ابر بسپارید. مانند این که اطلاعات خود را با یک بالن به ابرها سپرده‌اید. این ابزار برای افرادی که از سیستم‌عامل گوگل کروم استفاده می‌کنند نیز بسیار کاربردی است. به شکلی که هرگاه بخواهند اطلاعاتی را ذخیره کنند، به جای ذخیره، مستقیم در دیسک سخت، آن را به شکل مستقیم به فضای ابری خود ارسال می‌کنند که از تمامی خدمات ابری مشهور نیز پشتیبانی به عمل می‌آورد. برای مثال اگر در حال مشاهده یک آلبوم تصویری هستید، می‌توانید روی آن کلیک راست موس را فشرده و در فهرست نمایش داده شده، از طریق گزینه «ذخیره پیوند در...» آن را در خدمت ابری رایگان و یا پولی مورد نظر خود ذخیره کنید.■

یوتیوب را به صورت پنجره‌های معلق تماشا کنید Floating YouTube

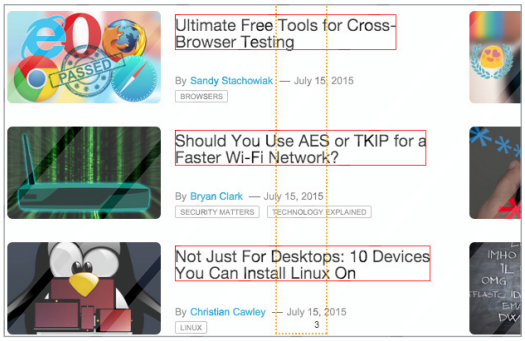
یوتیوب از آن دسته از خدمات برخط هم‌رسانی ویدیو است که هر چه خدمات مشابه ایرانی برای آن ایجاد شود، هرگز نمی‌توانند در مقابل امکانات و سطح خدمات و کیفیت آن دوام آورند، مگر با جلوگیری از دسترسی به این خدمت در داخل کشور که باعث اجبار کاربران برای مراجعه به خدمات دیگر خواهد شد. یوتیوب خدمتی است که توسط شرکت گوگل خریداری شده و هم‌اکنون نیز توسط گوگل پشتیبانی می‌شود و میلیون‌ها ویدیوی آموزشی، علمی، سرگرم‌کننده و جالب به زبان‌های مختلف دنیا در آن قرار دارد. تمامی ویدیوهای یوتیوب دارای کیفیت‌های مختلفی هستند که فراخور سرعت اینترنت، می‌توان یکی از این کیفیت‌ها را برگزید. در سال‌های اخیر قابلیت مشاهده ویدیو با کیفیت چهارهزار «4K»و نرخ‌فریم‌های بالاتر از سی فریم بر ثانیه نیز به یوتیوب افزوده شده‌است که برتری خاصی به این خدمت هم‌رسانی ویدیوی اینترنتی گوگل بخشیده است. با وجود این، مشاهده ویدیو در این خدمت از روش‌های مختلفی میسر است. چه با استفاده از ابزار ارزان قیمت کروم‌کست این دسترسی انجام شود، چه از طریق مرورگر اینترنتی، کیفیت این خدمت هم‌رسانی ویدیو تقریبا یکسان و راضی‌کننده است. نرم‌افزار کاربردی آن در اندروید و آی‌اواس قابلیت خوبی را برای کاربران فراهم می‌آورد که در آن می‌توان یک ویدیو را به صورتی کوچک در پایین و سمت چپ قرار داد. هر چند این قابلیت، امکان بسیار محدودی است اما با نصب افزونه‌ای در مرورگر اینترنتی گوگل کروم، قادر خواهید بود که ویدیوهای نمایش داده شده در یوتیوب را به شکل پنجره‌های معلق مشاهده کنید. کاربرد این افزونه برای افرادی است که به مشاهده ویدیو پرداخته و در عین حال کارهای دیگر را نیز در کنار تماشای آن ویدیو انجام می‌دهند.■

اجبار برای گشودن سربرگ‌ها در حالت پس‌زمینه Force Background Tab



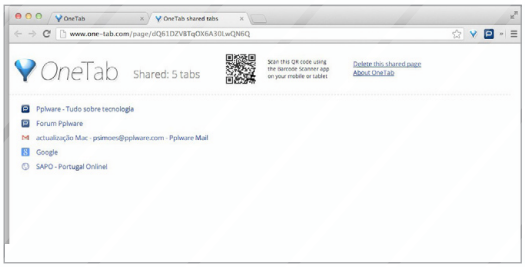
برخی مواقع ممکن است برای شما نیز پیش‌آمده باشد که در یک پیام رایانامه که برایتان ارسال شده، چندین پیوند به دیگر پایگاه‌های اینترنتی قرار دارد و شما مجبورید با کلیک بر هر یک از پیوندها، به اطلاعات این پیوندها دسترسی داشته باشید. در این مواقع شما هنوز خواندن آن پیام را به پایان نرسانده‌اید، ولی می‌خواهید پیوندهای فوق در سربرگ‌های جداگانه‌ای نمایش داده شوند. هم‌چنین در مواقعی که یک مطلب را مشاهده می‌کنید که در آن چند پیوند جالب قرار دارد، با کلیک روی آن، به آن پایگاه اینترنتی و یا صفحه موجود در پیوند منتقل خواهید شد و سربرگ جدید مقابل شما قرار گرفته و تمرکز شما را برای مطالعه آن مطلب به هم می‌ریزد. برای رفع چنین مشکلی، افزونه‌ای ساده در گوگل کروم قرار دارد که می‌تواند تمامی پیوندهای باز شده در سربرگ جدید را به صورت ساکت و در پس زمینه اجرا کند. به این شکل، آن سربرگ‌ها در سربرگ جدید باز می‌شوند؛ اما تا زمانی که نخواسته‌اید، به آن سربرگ منتقل نخواهید شد.■

گشودن چندین پیوند در یک حرکت Linkclump



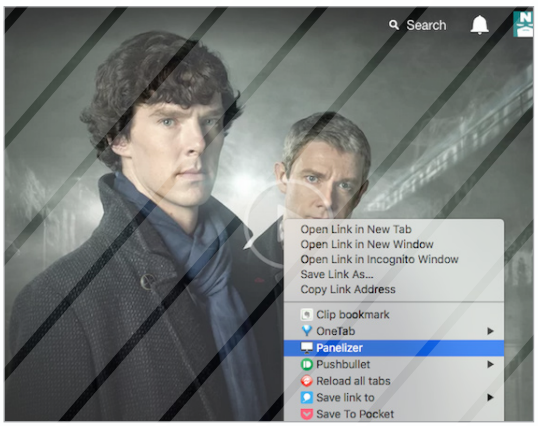
در برخی مواقع شاید نیاز داشته باشید، چندین پیوند را به آسانی و با یک کلیک راست در سربرگ‌های مختلف باز کنید. برای مثال چند عدد پیوند برای بارگیری چندین فایل موسیقی در یک صفحه قرار دارد که شما می‌خواهید به راحتی و با یک کلیک تمامی آنان را بارگیری کنید. برای این کار تمامی پیوندها را با کلیک راست موس انتخاب کرده و پس از این که در اطراف پیوندها کادر قرمز رنگی کشیده شد، موس را رها کنید تا تمامی پیوندها در سربرگ‌های مختلف باز شوند. برای دریافت این افزونه نیز به صفحه افزونه دراین پیوند به پایگاه اینترنتی افزونه‌های گوگل کروم مراجعه کرده و با کلیک روی دکمه آبی رنگ نصب افزونه‌ها، آن را نصب کنید.

نمایش تمامی سربرگ‌ها در یک سربرگ OneTab



هنگامی که چندین سربرگ در مرورگر اینترنتی به صورت هم‌زمان در حال اجرا باشند، معمولا سامانه دچار کندی و اختلال می‌شود. به طوری که کارایی و بازدهی آن تا مقدار زیادی کاهش خواهد یافت. در این مواقع به دلیل بارگذاری موارد متعدد مانند تصاویر و... در حافظه اصلی رایانه، حافظه پر می شود. با استفاده از افزونه یک سربرگ «OneTab»، هرگاه سربرگ‌های زیادی در مرورگر کروم در حال اجرا بود و کارایی سامانه تحت تأثیر قرار گرفت، می‌توانید روی نقشک این ابزار در نوار ابزار کلیک کنید تا تمامی سربرگ‌های درحال اجرا به یک‌باره بسته شده و تنها یک سربرگ که فهرستی از پیوندها به آن سربرگ‌ها است، ایجاد شود. این کار باعث می‌شود سربرگ‌های در حال اجرا در مرورگر بسته شده و حافظه اصلی خلاص شود و هم‌چنین نشانی سربرگ‌ها به همراه عنوان آن‌ها برای استفاده‌های بعدی حفظ شود.

تماشای ویدیو به صورت پنجره‌های معلق Panelizer



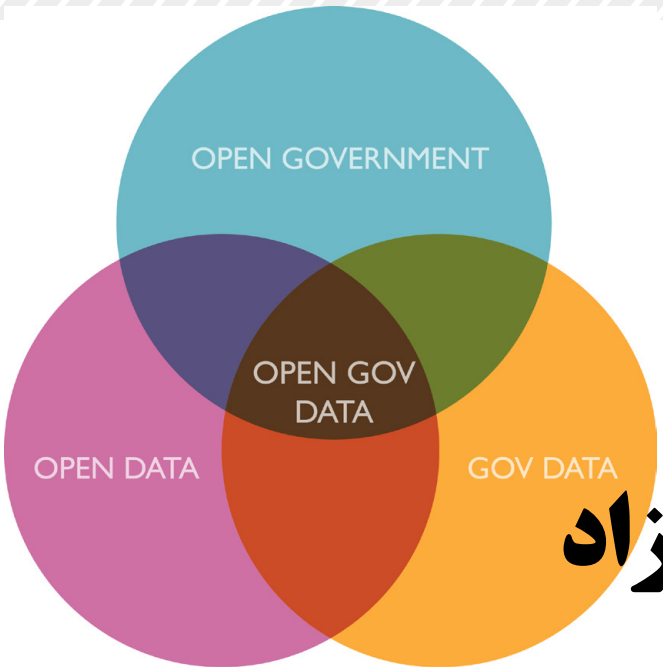
در یکی از افزونه‌های گفته شده در این مطلب، افزونه‌ای را معرفی کردیم که برای نمایش ویدیوهای یوتیوب به صورت معلق و در پنجره‌های کوچک کاربرد داشت. حال اگر بخواهید ویدیوها و موسیقی‌های برخی خدمات برخط مانند نت‌فلیکس را در کادری معلق به صورت یک اعلان مشاهده کنید، می‌توانید افزونه‌ای را نصب کنید که ویدیو را در پایین سمت راست نمایش می‌دهد. برخلاف افزونه مورد استفاده برای معلق کردن ویدیوهای یوتیوب، افزونه پنلایزر «Panelizer» قادر است ویدیو را خارج از چارچوب مرورگر نیز به نمایش بگذارد.

ابزار فوق، با خدمات مختلفی قابل استفاده است. با استفاده از این افزونه به راحتی می‌توانید ویدیوها را در یک کادر معلق نمایش داده و به دیگر کارهای خود مشغول شوید. برای دریافت این افزونه باید از طریق این پیوند به پایگاه اینترنتی افزونه‌های گوگل کروم شده و با کلیک روی دکمه آبی رنگ بالای کادر به نمایش درآمده، افزونه را نصب و استفاده کنید. برای استفاده از افزونه، ابتدا وارد پایگاه اینترنتی و یا صفحه دارای ویدیوی مورد نظر شده و با کلیک راست کردن روی ویدیو و مشاهده فهرست، روی گزینه «Panelizer» کلیک کنید. این کار باعث نمایش ویدیوی فوق به صورت کادری معلق خواهد شد.

تغییر تصویر پس‌زمینه سربرگ

اگر یک کاربر اینترنت فعال و همواره برخط هستید و در اکثر خدمات اینترنتی فعالیت می کنید، ممکن است برخی مواقع در استفاده از مرورگر و گشت و گذار در دنیای اینترنت با کمی کسالت و خستگی مواجه شوید که حوصله شما را سر خواهد برد. در این مواقع، انجام برخی تنظیمات ظاهری در مرورگر و سفارشی کردن آن بسیار مناسب خواهد بود. با وجود این اگر تصاویر قالب‌های کروم و ... برایتان خسته کننده می‌شود، بهتر است از افزونه‌ی درج تصویر برای سربرگ‌ها «tab pics» استفاده کنید که با اتصال به اینترنت، تصاویر جدید را بارگیری و به‌عنوان تصویر زمینه در سربرگ‌ها از آن‌ها استفاده می کند. در هربار استفاده، تصویری جدید از اینترنت بارگیری می‌شود که باعث تنوع در ظاهر مرورگر خواهد بود.

کسب و کار



اهمیت چرخش آزاد اطلاعات دولتی

نویسنده:
دکتر جولیا گیلدن

سازمان ملل متحد در تحقیقات و پژوهش‌های اخیر که از حدود ۵ سال پیش تا حال، نیز به این نکته اشاره داشته است که اطلاعات آزاد می‌تواند به گشودن عرضه، تولید و انتشار دانش محلی جهانی شود. لازم به ذکر است که گردش آزاد اطلاعات دولتی و اطلاعات باز/آزاد برای تمامی دولت‌ها حیاتی است. موسسه اطلاعات باز (Open Information Foundation) براساس گزارش شرکت مک‌کینزی «۷ بخش به تنهایی می‌توانند ۳ میلیارد دلار در سال ارزش افزوده را به واسطه اطلاعات آزاد ایجاد کنند». «آزادی اطلاعات» و «حق دسترسی به اطلاعات»، به این معنی است که اکثر شهروندان توانایی دسترسی به بخش اعظمی از اطلاعات دولتی غیرمحرمانه و حساس را داشته باشند، که برای شفافیت دولت و از بین بردن فساد اقتصادی امری حیاتی است. چندان که آزادی اطلاع‌رسانی و دسترسی به اطلاعات در قرون جدید و پسارنسانس به «رکن اصلی دموکراسی» تعبیر می‌شود.

بی‌هیچ شک اطلاعات باز و آزاد، سیاست عمومی بسیار مهمی است و دغدغه اصلی کشورهای در حال توسعه قرار گرفتن در مسیر صحیح توسعه و عدالت و شفاف‌سازی است؛ از سوی دیگر آمار و پژوهش‌های رخداده در سازمان‌های بین‌المللی، بیانگر آن است که از مهم‌ترین عوامل در رشد و توسعه کشورها، «حکمرانی خوب» است. چنان که حکمرانی خوب را بنیان توسعه خوانده‌اند. حکمرانی خوب از جمله مباحث تازه‌ای است که در دو دهه اخیر توجه پژوهشگران و محافل علمی و بین‌المللی جهان را به سوی خود جلب کرده است. به این دلیل که حکمرانی خوب به چرخش آزاد اطلاعات بستگی دارد و حکمرانی خوب نیز از ملزومات توسعه است، امروزه همه صاحب‌نظران به این موضوع معترف هستند که باید افراد به اطلاعات دولتی دسترسی بیشتری داشته باشند و مسائل مالی و سیاسی در دولت شکلی شفافتر به خود گیرد.

در تحقیقاتی جدید در کمیسیونی در اروپا بر روی پژوهش و مطالعه‌ای کار می‌شود که با استفاده از این پژوهش، بتوان چرخش آزاد شکاف‌ها را بین دولت و اصحاب عرصه اطلاع‌رسانی کاهش دهند و درک متقابل هر یک را نسبت به موضع بالاتر ببرند. بر این اساس آن‌ها افراد را تشویق می‌کنند تا اطلاعات خود را به وسیله نرم‌افزارهای آزاد / متن‌باز منتشر نمایند. همچنین دولت‌ها نیز می‌توانند با استفاده از نرم‌افزارهای آزاد و یا حتی تولید نرم‌افزارهای دولتی به شکل متن‌باز به آزادی چرخه گردش اطلاعات، کمک کنند. این پژوهش و مطالعه تا به حال در ۱۴۰ شهر از ۶ کشور جهان برای گسترش و توسعه و انتقال دید مناسب درباره آزادی اطلاعات، مشغول اطلاع رسانی و پژوهش‌های محلی هستند.

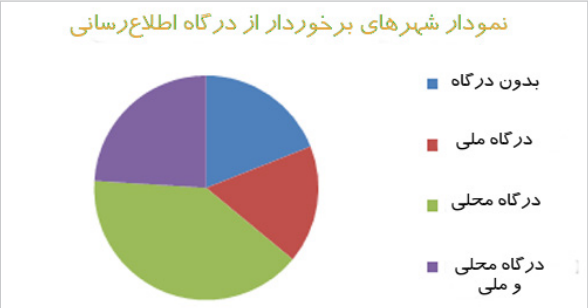
صاحب‌نظران به این موضوع معترف‌هستند که باید افراد به اطلاعات دولتی دسترسی‌بیشتری داشته باشند و مسائل مالی و سیاسی در دولت شکلی شفافتر به خود گیرد.

نکات کلیدی به دست آمده از سطح بلوغ شهرهای مورد پژوهش در زمینه چرخش آزاد اطلاعات را می‌توان به موارد زیر خلاصه کرد؛

- ۱۷ درصد از آنان هیچ گونه آشنایی و ارتباطی با اطلاعات آزاد نداشته و با این موضوع کاملا غریبه بودند
- ۲۴ درصد آنان تجارب کمی در مورد اطلاعات آزاد داشتند؛ مانند برخی اطلاعات خاص و نه اطلاعات شهری و یا اطلاعات سیستماتیک و مهم
- ۴۷ درصد از این شهرها از تجارب نسبتا متوسط و خوبی در زمینه اطلاعات آزاد و چرخش اطلاعات برخوردار بودند. با این حال سیاست مشخصی در قبال انتشار اطلاعات و یا قوانین مرتبط با اطلاعات آزاد در آنان وجود نداشت و همچنین به صورت شفاف مشخص نبود که چه مواردی باید محرمانه باشد.

۱۲ درصد از شهرها نیز تجربیات پیشرفته و بالایی را در این زمینه دارا بودند؛ از سیاست‌های شفاف و درگاه‌های اطلاع‌رسانی اینترنتی شهری، بسیار خوبی برخوردار بودند و به طور کلی در شرایط مطلوبی قرار داشتند.

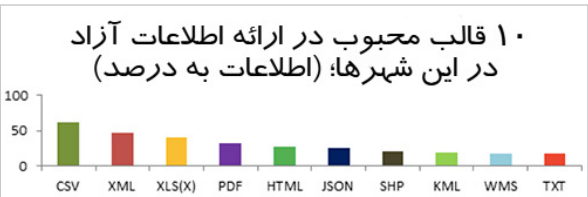
همچنین بر اساس پژوهش صورت گرفته در این پژوهش و مطالعه مشخص شد که ۱۰ درصد از شهرها اصولا دارای سامانه اطلاع‌رسانی یا پایگاه اینترنتی و حتی درگاه منسجم ارتباطی برای ارائه اطلاعات دولتی و شهری نبوده و در زمینه ارائه اطلاعات آزاد و



چرخش اطلاعات، پیشرفت خاصی نداشته‌اند. همچنین ۱۷ درصد از آنان به یک درگاه ملی اطلاع‌رسانی آزاد برخوردار بودند و ۴۴ درصد از آنان از درگاه محلی برای ارائه اطلاعات آزاد برخوردار بوده و ۲۹ درصد از آنان نیز هم دارای درگاه محلی بودند و هم درگاه ملی.

در پایان نیز بر اساس قالب انتخابی برای انتشار آمار و اطلاعات دولتی و قرار دادن آن در اختیار مردم، حدود ۷۷ قالب مجزا در حال استفاده است که برخی از این قالب‌های انحصاری و برخی دیگر نیز آزاد / متن‌باز بودند. برخی از این قالب‌ها نیز قالب‌های تخصصی برای برخی امور خاص علمی و آماری بودند. پرکاربردترین قالب را می‌توان قالب CSV که نوعی قالب صفحه گسترده است، دانست که قالب ۶۲ درصد از پرونده‌های ارائه شده را شامل می‌شد. قالب بعدی نیز قالب XML بود که یک زبان نشانه‌گذاری دیگر مانند HTML است. یکی دیگر از قالب‌های پرکاربرد دیگر که برای نگاهداری اسناد و جداول صفحه‌گسترده به کار می رود همان پرونده‌های برنامه اکسل مایکروسافت یعنی XLS بود که با وجود انحصاری بودن در اکثر نرم‌افزارهای تحت‌وب، گوشی‌هوشمند و یا آزادی چون لیبره‌آفیس قابل ایجاد و مشاهده است

اکثر شهرها اطلاعات خود را با استفاده از یک یا دو سه قالب خاص منتشر کرده‌اند. بااین‌حال در رسم زیر می‌توانید نموداری از ۱۰ قالب محبوب در ارائه اطلاعات آزاد در این شهرها را مشاهده نمایید؛



براساس پژوهش این گروه مشخص گشته است که حداقل نیمی از شهرها درک صحیحی از مزایای مرتبط با استفاده از برخی قالب‌های مورد استفاده خود ندارند؛ براین‌اساس اکثر شهرهای مورد پژوهش توسط این پژوهش و مطالعه نشان می‌دهد که به جز معدود شهرهای پیشرفته اکثر این شهرها از قالب‌های مناسب برای انتشار اطلاعات استفاده نمی‌کنند. بر اساس استانداردهای جاری بین‌المللی برای انتشار اطلاعات به صورت آزاد، اطلاعات باید از قالب‌های خاصی استفاده نمایند که برای محتوایی که قرار است توسط آن ارائه شود کاملا مناسب باشند و علاوه‌براین موضوع بهتر است متن‌باز و آزاد باشند. زیرا اگر وابسته به فناوری و نرم‌افزار خاصی باشند، قابلیت انتقال کمتری خواهند داشت؛ به شکلی که اگر نرم‌افزار ایجادکننده آن پرونده تغییر یافت به راحتی بتوان به ویرایش و مشاهده آن داده‌ها پرداخت.

همان طور که اشاره‌شده استفاده از قالب‌های آزاد یکی از بهترین روش‌های انتشار محتواست که تضمین می‌کند اطلاعات عرضه شده بدون وابستگی به نرم‌افزار خاصی قادر به مشاهده و ویرایش باشند. بدون این ویژگی عملا بخشی از آزادی چرخش اطلاعات محدود به افراد خاصی خواهد شد که دسترسی به نرم‌افزار مورد استفاده برای ایجاد پرونده، داشته و می‌توانند از آن استفاده نمایند. بنابراین بهتر است برای ارائه اطلاعات از نرم‌افزارهای آزاد / متن‌باز استفاده شود.

در ابتدا اکثر افراد حتی در گروه پژوهشی مورد اشاره از مدل داده‌های پیوندی متن‌باز (آزاد) LOD حمایت می‌کردند که داده‌های مورد نیاز را به صورت مجموعه‌ای به هم پیوسته ذخیره می‌نمود. همچنین در این مدل از روش آردی‌اف «RDF» برای ذخیره و بازیابی معنای قابل پردازش توسط ماشین بکار می‌رفت؛ به این دلیل که این مدل با زبان اکس‌ام‌ال نوشته می‌شود؛ در تمامی ویرایشگرهای متنی قابل ویرایش است. بااین‌حال به دلیل پیچیدگی این مدل و سختی یادگیری ایجاد چنین پرونده‌هایی، این مدل دیگر توسط افراد غیرحرفه‌ای و عادی استفاده نمی‌شود، به این‌خاطر این مدل رواج زیادی در شهرهای مورد پژوهش ندارد.

در طی سالیان مختلف استفاده از قالب زبان نشانه‌گذاری اکس‌ام‌ال در حال افزایش است. این زبان قدرت بالایی در نمایش اطلاعات مختلف به صورت‌های مختلف دارد که برتری خاصی به آن بخشیده است. از دیگر مزایای آن می‌توان به امکان ارائه آسان و راحت آن در درگاه‌های اینترنتی اشاره کرد که حتی می‌توان آن را با پایگاه اینترنتی و دیگر مطالب به صورت درونی ارائه داد. این قالب کاربردهای فراوانی دارد و از ساختار بسیار ساده‌ای نیز برخوردار است به عنوان مثال اکثر برنامه‌های گرافیکی از ساختار اکس‌ام‌ال برای چیدمان عناصر بصری نرم‌افزار خود استفاده می‌کنند؛ مانندِ گوگل اندروید که از اکس‌ام‌ال برای طراحی عناصر گرافیکی نرم‌افزارهای کاربردی خود بهره می‌برد.

اخیرا قالب جدید و متن‌بازی با نام سی‌اِس‌وی «CSV» یا (مقادیر جداشده با ویرگول) مورد استفاده قرار می‌گیرد که داده‌های آن با ویرگول از هم جدا می‌شوند. پرونده‌هایی که با استفاده از چنین روشی ایجاد شده‌اند را می‌توان با استفاده از نرم‌افزارهای اکسل و کلک (calc) لیبره‌آفیس باز نمود. سی‌اِس‌وی را می‌توان یکی از محبوب‌ترین قالب‌ها برای ارائه محتوای صفحه‌گسترده دانست که به صورت گسترده توسط شهرهای مورد پژوهش نیز استفاده شده است. از مزایای آن نسبت به اکس‌ام‌ال می‌توان به سادگی کار با آن اشاره کرد؛ علاوه‌براین بدون نیاز به نرم‌افزار خاص و تنها با یک ویرایشگر ساده نیز قابل ویرایش است. همچنین زبان‌هایی مثل پایتون و پِرل نیز از توابع درونی خاصی برای کار با این پرونده‌ها برخوردارند که توسط نرم‌افزارهای نوشته شده در این برنامه‌ها نیز می‌توان داده‌های ذخیره شده در این قالب را پردازش کرد.

به‌طورکلی از تمام موارد گفته شده و با وجود مدل ارائه داده پیشرفته LOD و آردی‌اف که توسط اکثر شهرهای مورد پژوهش استفاده نشده‌اند می‌توان گذاره‌های زیر را نتیجه گرفت؛

دارندگان اطلاعات شهری سادگی و راحتی استفاده از قالب سی‌اِس‌وی را ترجیح می‌دهند

طرف‌داران اطلاعات باز همواره از سی‌اِس‌وی در مقابل مدل‌هایی مثل آردی‌اف، پشتیبانی کرده و معتقدند اطلاعات آن وضوح بیشتری دارند.

گروه درحال‌رشد و بزرگی از برنامه‌نویسان و توسعه‌دهندگان ترجیح می‌دهند به جای استفاده از بانک‌های اطلاعاتی و غیره، از قالب سی‌اِس‌وی استفاده نمایند که انعطاف بیشتری را برای کار با داده‌ها، فراهم می‌کند.

شهروندان عادی آرزو دارند از اطلاعات در قالب سی‌اِس‌وی استفاده نمایند؛ زیرا چنین قالب‌هایی برای استفاده توسط آنان بسیار راحت‌تر و مناسب‌تر است تا استفاده از ساختار پیشرفته اکس‌ام‌ال و آردی‌اف.

استفاده گسترده از اطلاعات ساده و آسانی چون سی‌اِس‌وی، نشان می‌دهد که قالبی که اطلاعات را با پیچیدگی کمتری در خود نگاه داشته و نمایش دهد، بسیار بیشتر از قالب‌های پیشرفته مورد استفاده قرار می‌گیرد. همچنین کاربران حرفه‌ای و توسعه‌دهندگان نیز برای انعطاف و راحتی بیشتر در هنگام برنامه نویسی چنین قالب‌هایی را به دیگر قالب‌ها ترجیح می‌دهند.■

داده‌های باز بیشتر
برای کاربران بیشتر...

۴۰+

کشور دنیا دارای دولت با بسته داده‌های باز هستند

۹۰,۰۰۰+

مجموعه اطلاعاتی باز در سایت data.gov برای امریکا وجود دارد

۱,۴۰۰,۰۰۰

بازدید از سایت «اطلاعات باز» انگلستان در تابستان ۲۰۱۳ صورت پذیرفته

۱۰۲ شهر

در روز هکاتون بین‌المللی «اطلاعات باز» ۲۰۱۳ مشارکت کردند

۱,۰۰۰,۰۰۰+

مجموعه اطلاعاتی در سراسر دنیا توسط دولت‌ها باز شده‌اند

تا سال ۲۰۱۳

...ارزش بیشتری را می‌تواند به همراه آورد

+ ۵۰٪

سهم مصرف‌کننده از ارزش بالقوه اطلاعات باز

+ ۱۰۰,۰۰۰

نرم‌افزار پزشکی، سلامتی و تناسب اندام برای تلفن‌های هوشمند

۳۵ ساعت در سال

صرفه‌جویی برای مسافران با آگاهی از تغییر زمان‌بندی‌ها براساس اطلاعات باز

۳,۰۰۰,۰۰۰,۰۰۰ واحد

گاز دی‌اکسیدکربن برای ساختمان‌هایی که می‌توانند توسط اطلاعات باز شناسایی شوند، کم‌تر منتشر می‌شود

۳,۰۰۰,۰۰۰,۰۰۰,۰۰۰ دلار

تخمین ارزش بالقوه اطلاعات باز در هفت زمینه است

۱۱ نکته برای هنگامی که استارت‌آپ شما با بازار جدید مواجه می‌شود

چگونه کسب‌وکار موفقی را راه‌اندازی کنیم

نویسنده:
اسکات گریر

جلب نظر و مجذوب کردن مشتریان از آن دسته چالش‌هایی هستند که اکثر فعالین اقتصادی با آن دست و پنجه نرم می‌کنند. اما حتی اگر شما یک Business Man کاملا موفق باشید، نیاز به کشف بازارهای جدید و مشتریان جدید خواهید داشت. حسی که در این مواقع به شما دست خواهد داد حس شروع دوباره همه این مراحلی است که طی کرده‌اید. برای صرفه‌جویی در هزینه و زمان، من از چند تن از کارآفرین‌های YEC (Young Entrepreneur Council) سؤالاتی پرسیدم که چه چیزی را فراتر از هر چیزی در اولویت قرار داده اند، زمانی که می‌خواستند، مدل خود را مبتنی بر مشتریان جدید تغییر دهند.

بهترین جواب‌های آنان در زیر آمده است:

بازخورد مشتری:

وقتی درحال‌توسعه مدل خود هستید، ضروری است که شما باعلاقه و به‌سرعت به نیازهای مشتریان خود توجه کنید. چیزهایی که شما میدان‌دار ثابت آنان هستید؛ زمانی که وارد میدان جدید می‌شود ممکن است دیگر این‌طور نباشد.

ضروری است که در روزها و یا ماه‌های اول سؤالات و فرم‌هایی را به مشتریان بدهید تا از آنان اطلاعاتی را جمع کنید که به‌صورت بالقوه برای گرفتن بازخورد مفید هستند. این بازخوردها را تحلیل کنید و بسنجید که در کجای کار ایستاده‌اید؛ این اعمال باعث می‌شود که به‌سرعت اشکالات کار را رفع کنید و یا چاه‌هایی را پرکنید، که در صورتی که شنونده خوبی نباشید؛ ممکن است درون آنان بیفتید.

دیدی کلی از مشتری نهایی داشته باشید، B2B یا B2C کارهایی اشتباه هستند مسائلی که باعث می‌شود پول خود را دور بریزید یا منجر به تلاشی بی‌فایده در پایین جاده شود.

در روزها و یا ماه‌های اول سؤالات و فرم‌هایی را به‌مشتریان بدهید تا از آنان اطلاعاتی را جمع کنید

پتانسیل شراکت:

قبل از این که وارد بازار جدید از مشتریان شوم، تحقیقات خود را انجام می‌دهم، سپس به یکی از شرکا و یا وابستگان تماس می‌گیرم تا از آنان مشورت بگیرم.

این کار را با دعوت کردن از آنان برای یک ناهار انجام می‌دهم. من از آنان تمامی خوب و بد و جزییات کار را جویا خواهم شد.

این به یکی از بزرگ‌ترین روش‌های من تبدیل شده است تا به استراتژی خود شکل دهم. به طور کلی اگر شریکی وجود داشته باشد که پلی بین خلاءهای بازار جدید و قدیم من ایجاد کند، من تلاش خواهم کرد تا آن را کاملا آشکار کنم.

من معتقد هستم که مشتریان هم از این همگام بودن لذت خواهند برد و این همیشه مانند آن است که گویی برند مورد علاقه آنان به آن‌ها انسانی برخورد کرده و دوست آنان هستند. اگر آنان با شما جور شوند، آن وقت است که به تک تک کلمات شما باور خواهند داشت.

بررسی تحقیقات اینترنتی:

مطمئن شوید که شما در مورد بازار مناسبی تحقیق می‌کنید، از سایت‌هایی مثل آمازون استفاده کنید وارد دسته‌بندی‌های مشابه شوید و ببینید که دیگران در مورد محصولاتی که فعلا در بازار هستند چه نظراتی دارند. مثلا مشکلات و گرفتاری که آنان با محصولات دیگر دارند و یا چیزهایی که در مورد آن محصولات دوست دارند.

همچنین چک کنید که معمولا مردم برای چه چیزهایی پول پرداخت کرده‌اند و چه محصولاتی فروش بیشتری داشته‌اند، آمازون (در ایران دیجی‌کالا یا ...) و Yelp سایت‌های خوبی برای انجام این دست تحقیقات هستند.

رقابت:

شاید هرگز نیاز نباشد تا به ارزیابی کلی از رقبا یا ظرفیت اسمی بازار بپردازید. مهم است که بدانید رقبای شما تا چه حد قدرتمند هستند، چه‌کارهایی را به خوبی انجام می‌دهند، چه چیزهایی را به خوبی انجام نمی‌دهند، خودتان را از بازار چگونه می‌توانید افتراق دهید و این‌که بازار هدف چه قدر بزرگ است.

همچنین شما نباید چنین تصور کنید که قرار است به شکلی خیره‌کننده از رهبر فعلی بازار بزرگ‌تر شوید.

مشتریان فعلی شما:

کاربران و مشتریان فعلی در حال حاضر اعتماد لازم را به برند شما دارند، مطمئن شوید که هرگز آنان را در هنگام انجام این تغییرات ناامید نکنید (بیگانه نشمارید) زیرا این مشتریان سابق شما هستند که باعث شده‌اند که چیزی که امروز هستید؛ باشید، با این حال سعی کنید خود را در چیزی که هستید ایزوله نکنید!

پتانسیل تغییرات:

هنگامی که زمان آن فرا می‌رسد تا مدل خود را مبتنی بر مشتریان جدید بنا نمایید، باید روی این موضوع تمرکز کنید که شما قرار است برند جدید را معرفی کنید که بتواند مخاطبین جدید را تحت تأثیر خود قرار دهد. این کار ممکن است برای سازمان‌ها که قصد تغییر هویت دارند چالش‌برانگیز باشد که هویت زیبای خود را حفظ کنند و همین‌طور بتوانند موارد جدید و قابل قبولی برای عرضه داشته باشند.

هنگامی که شما مراحل تکامل به سمت بدل شدن به یک برند و هویت جدید را طی می‌کنید، نیاز دارید تا نام تجاری خود را در موضوعات جدید جای بیاندازید، از بازار جدید که قصد ورود به آن را دارید، یک مصاحبه از افراد داشته باشید تا درک درست و جامعی از نیازها و ترجیحات آنان داشته باشید. این به شما کمک خواهد کرد که اگر شرایط در بازار جدید بحرانی شد بتوانید از پس بحران پیش‌آمده برآیید.

نیازها را از نو تعریف کنید:

شما باید وقت بیشتری را صرف کنید و تلاش بیشتری نمایید تا به مشتری جدید دست یابید. با تلاشی بسیار بیشتر از زمانی که باید برای جلب مشتریان فعلی خود را انجام دهید. مردم معمولا خیال می‌کنند، که تغییر بازار و کسب به هدفی جدید بسیار ساده است؛ اما اغلب فراموش می‌کنند که تلاش برای دستیابی به بازارهای جدید، همیشه برای آنان و اطرافیانشان تلاش بزرگی بوده است.

درباره اولین بازاری که تجارت شما در آن کلید خورد فکر کنید. این کار به همان اندازه نیاز به تلاش دارد که هنوز برند شما شناخته شده نبود. در هر حال شما نیاز دارید تا نیازهای تجاری خود را از نو تعریف کنید.

دیدگاه از زمینه‌های دیگر :

روش معمول برای تحقیق در مورد یک مارکت به‌راحتی با نگاه کردن به روند مشتریان در زمینه کاری شما و پرسش سؤال از مشتریان فعلی قابل انجام است، که چه چیزی را می‌خواهند در آینده شاهد باشند. به‌جای آن که خود را طبقه‌بندی کنید! تلاش کنید تا دید مناسبی، از موضوع کاری خود به وسیله مطالعه دیگر شرکت‌ها در زمینه مشابه خود به‌دست آورید. از تجربیات آنان بیاموزید و از نحوه ورود آنان به بازار جدید و راه‌حل‌های آنان کسب تجربه کنید.

تجزیه و تحلیل کامل بازار:

شما نیاز دارید که تشخیص دهید که چه چیزی واقعا نیاز است، این کار را با مطالعه و تحقیق دقیق و جمع‌آوری اطلاعات به دست خواهید آورد. شما نباید این تصمیمات را سرسری بگیرید.

یک مطالعه درست می‌توان از چالش‌هایی که ممکن است برای شما اتفاق بیفتد پرده‌برداری کند. پس شما می‌توانید آنان را پیش‌بینی کنید و محصول و کمپین خود را به‌صورت مناسبی برنامه‌ریزی کنید.

پیام‌های استراتژیک:

مطمئن شوید که پیام خود را درمورد محصولات به‌درستی به مخاطبین انتقال می‌دهید. اگر شما می‌خواهید که بین محصول جدید که در بازار قدیم عرضه می‌کردید و محصولات جدید در بازار جدید، تمایز قائل شوید؛ باید در پیام‌های شما این هدف و منظور کاملا روشن باشد.

توسعه یک محصول شمار را آماده می‌کند با این فرصت که بتوانید مشتریان فعلی را درباره رشد، مسیر شما و قدرت خلاقیتتان، هیجان زده کنید. بااین‌حال اگر موقعیت مناسبی را نتوانید فراهم کنید، موفق نخواهید بود.

تغییرات در فروش و خدمات:

به چالش‌های کلی در رابطه با فشارها بین مشتری و تیم خدمات چشم نپوشید. شما به افراد مشخصی آموزش داده‌اید تا محصول خاصی را به فروش برسانند یا با مشتریان خاصی مواجه شوند. حال زمانی که مدل خود را تغییر می‌دهید، باید خود را آماده کنید تا چرخه‌ی فروش، تغییرات محصولات و خدمات را به‌روز نمایید. حتی مجبور خواهید بود تا چرخه‌ی توزیع، فروش و یا خدمات پس از فروش را از نوع تنظیم نمایید.■

مهم است که بدانید رقبای شما تا چه حد قدرتمندهستند، چه‌کارهایی را به خوبی انجام می‌دهند، چه چیزهایی را به خوبی انجام نمی‌دهند

تجربیات مدیرعامل ردهت از چندین سال مدیریت

نویسنده
برایان برنشورن

در خلال جشنی که برای انتشار کتاب «The Open Organization» توسط جیم وایت‌هارست (مدیرعامل شرکت ردهت) در حال انجام بود، بیش از ۴۵۰ نفر از رهبران کسب‌وکارهای کوچک و بزرگ و فعالین بازار در یک کنفرانس مجازی اینترنتی دور هم جمع شده بودند تا به سخنان مدیرعامل شرکت ردهت درباره مدیریت تغییرات سریع رخ داده در اقتصاد در آینده و حال صحبت نمایند.

شروع پرسیدن سوالات در ابتدا از جانب لیندا مرینوس صورت گرفت که سردبیر نشریه تجاری هاروارد است. سوال او این بود که «آیا برای نوشتن کتاب خود از نرم‌افزاری متن‌باز بهره جسته‌اید؟» وایت‌هارست سریعا پاسخ داد که «بله، ما کار را با استفاده از نرم‌افزار آزاد انجام دادیم، از لیبره‌آفیس که یک واژه‌پرداز محبوب متن‌باز است استفاده کرده‌ایم. و درحالی که میکروفون را با هیجان زیاد در دست خود می‌فشرد گفت: «و البته ماجراهای جالبی در این‌باره دارم».

سازمان‌های آزاد (باز) در عمل

مثل تمامی محصولات شرکت ردهت کتاب سازمان‌های آزاد هم ناشی از تلاش مشترک همه همکاران در ردهت است که به واسطه تجارب و بینش آنان شکل گرفته است. کتاب فوق را می‌توان چکیده‌ای از تجارب و آموخته‌های وایت‌هارست در عرض هفت سال مدیریت وی بر ردهت دانست. این کتاب چراغ راهی است برای مدیران دیگر موسسات مرتبط با مدل آزاد / متن‌باز که چگونه بتوانند با حفظ شایسته‌سالاری، اصول همکاری و شفافیت خود را برای رویارویی با قرن ۲۱ام آماده کنند؛ به‌طوری‌که وایت‌هارست معتقد است این کتاب در مورد مواردی است که او تا به حال آموخته است. همانند بخش‌های دیگر دنیای آزاد / متن‌باز این کتاب نیز برای نقد و بررسی در اختیار عموم بوده است؛ تا این که بالاخره به تدریج منتشر شود.

خوانندگان می‌توانند نظرات و پیشنهادات

بله، ما کار را با استفاده از نرم‌افزار آزاد انجام دادیم، از لیبره‌آفیس که یک واژه‌پرداز محبوب متن‌باز است استفاده کرده‌ایم

خـود را درمـورد هـر بخشـی از مطلـب در کتاب به واسطه قابلیت کامنت «Comment» لیبره‌آفیس بیافزایند؛ که به خوانندگان اجازه حاشیه‌نویسـی بر مطالب را خواهد بخشید. بـا این حـال زمانی کـه جیم وایت‌هارسـت می‌خواست آنان را چاپ کند، در مکان‌هایی که انتظار می‌رفت باشـند، حضور نداشـتند. آن‌هـا به بخـش زیرین صفحـات می‌افتادند ولی جیم وایت‌هارست می‌خواست آنان را به قسـمت‌های کناری و حواشی کتاب منتقل کند، درست همان طور که دلش می‌خواست و در هنـگام کار بـا صفحه‌نمایش مشـاهده می‌کرد.

بنابراین او ایده جدید خود را برای بهبود این قابلیـت در لیبره‌آفیـس با همـکاران و دیگر توسعه‌دهندگان در ردهت، در میان گذاشت که عرض چند هفته یکی از توسعه‌دهندگان نرم‌افزارهای آزاد در شـرکت ردهت کائولان مک‌نامـارا ایـن ایـده را بـرای نسـخه جدید لیبره‌آفیـس پیاده کرد. (به نظر می‌رسـد در دنیـای نرم‌افزارهـای آزاد / متن‌باز نیز حرف مدیران زودتر مورد توجه قرار می‌گیرد!)

جیم می‌گوید «در واقع دیگر چنین قابلیتی را در خـود نرم‌افـزار لیبره‌آفیـس داشـتیم، روزی فقـط یـک به‌روزرسـانی در (توزیـع) فـدورای خود مشـاهده کـردم، کـه در واقع همـان ویژگـی مورد انتظار من بـود!» با این حال جیم وایت‌هارسـت تنها کسی نبود که از ایـن قابلیـت منفعت می‌بـرد، بلکه پایگاه اینترنتی ZDNet نیز افزوده شدن این قابلیت را بـه عنـوان «قـدم حقیقـی و روبه‌جلـو» جشن‌گرفته بود.

قطعـا ایـن کتـاب از ایـن جهت بـرای جیم وایت‌هارسـت مهم است؛ زیرا او داستان‌های زیادی را در مورد آموخته‌هایش در آن آورده اسـت که وی امیدوار بوده برای سازمان‌های آزاد (باز) مهم هستند. (براساس این ایده‌ای که می‌گوید؛ پروژه‌های آزاد / متن‌باز الگویی را ارائـه می‌دهنـد که موفقیت سـازمان‌ها را تضمیـن می‌کنـد.) به صورت کلـی کتاب با هدف آموزش مدیران و دیگر رهبران نوشته شده است تا به آنان یادآوری کند، تا چه حد مدیریـت و ایده‌های سـنتی ناقص بوده و به طور فزاینده دیگر حرفی برای گفتن ندارند.

جیم وایت‌هارست می‌گوید «دنیایی که ما در حال اعمال مدیریت در آن هستیم، با دنیای قدیمـی و سـنتی دیـروز کـه در آن آموزش دیده‌ایم؛ کاملا متفاوت اسـت. اما پروژه‌های آزاد / متن‌باز موارد جدیدی را در چنته دارند؛ راه‌های جدید برای سازمان‌دهی، همکاری و رهبری یک تیم.» به عنوان مثال و در موردی که برای نرم‌افـزار لیبره‌آفیس صورت گرفته بود؛ جیم وایت‌هارسـت موفق شـد با فشـار بر کارکنان خـود بتواند چنین قابلیتی را به نرم‌افـزار مجموعه دیگری اضافه کند. چنین امکانی برای نرم‌افزارهای انحصاری و غیرآزاد وجود نـدارد و فرد نمی‌توانـد به کدهای آن دسترسی داشته باشـد. بنابراین دانش افراد در یـک شـرکت می‌تواند برای شـرکت‌های دیگر نیز مورد استفاده قرار گیرد که منجر به همکاری بهتر شرکت‌ها نیز می‌شود.

بـا تمـام این تفاصیـل، جیم وایت‌هارسـت معتقـد اسـت در سـازمان‌های آزاد تصمیم‌گیری‌ها به صورت دموکراتیک نیست و برخـی موارد زور و قدرت کاملا بین اعضاء تقسـیم نخواهد شد و در ساخت و همکاری و تولیـد و راه‌انـدازی یـک پـروژه، اولویت با کسـانی خواهد بود که در این مسائل زودتر حاضر بوده‌اند. جیم وایت‌هارسـت می‌گوید: «صادقانه بگویم، صدای برخی افراد بیشتر از، دیگر افراد شنیده می‌شود.» که این موضوع بر اسـاس شـهرت اسـت. اکثر افراد شـهرت خود را با استفاده از گفتگو و مکالمه به دست آورده‌اند؛ جیم وایت‌هارست هم معتقد است که مکالمه و گفتگـو در دنیای آزاد / متن‌باز یکـی از دلایـل شـهرت افراد بـوده و یکی از اساسـی‌ترین روش‌های جدا کردن سره را از ناسـره به شـمار می‌رود. به طور کلی یکی از روش‌هایی که افـراد در جامعه آزاد / متن‌باز محبـوب و مشـهور می‌شـوند نوع و سـطح ارتباط آنان با افراد جامعه است که منجر به این محبوبیت خواهد شد.

جیم وایت‌هارست با خنده می‌گوید «ما یک سازمان کوم‌بایا «kumbaya» نیستیم، روش سـنتی طوفان‌مغزهـا، جایی که افـراد همه ایده‌هـای خود را مطـرح می‌کنند تا یکی از آنان برگزیده شـود دیگـر روش خوبی برای تصمیم‌گیری نیسـت؛ بلکه مباحث ساینده و مکالمات هسـتند کـه باعث اتخاذ تصمیم درسـت می‌گردنـد.» وی می‌افزاید که شـما بـه پوسـتی کلفت هـم احتیاج داریـد؛ تا به نظـرات و ایده‌های آنان توجه کنید و از بین ایـن نظـرات و ایده‌ها، نظرات خوب و برتر را انتخاب نمایید. این موضوع نیازمند آن است کـه توجه و وقت بیشـتری را صرف دقت به نظرات دیگران نمایید. وی معتقد است، این روش، یکـی از روش‌های جدیدی اسـت که در پروژه‌های منبع‌باز از آن استفاده می‌شود. همچنین جیم وایت‌هارسـت اشاره می‌کند که شـما کدهایتان نیستید، از منظر او افراد برای پیشـرفت پـروژه خود باید احساسـات شخصی را کنار گذاشته و نظرات دیگران را به خوبی بشنوند و انتقادات را جدی بگیرند و اجازه ندهید انتقاد شما را به هم بریزد بلکه با روحیه ای شاد با آن مواجه شوید.

همچنیـن بـه عنوان یـک نصیحـت عرض می‌کنـم که کینه بـه دل نگیرید. عصبانی و نگران ماندن از یک اشتباه می‌تواند کار آتی شما را هم تحت تاثیر قرار دهد. اشتباه را از ذهن خود بیرون کنید . بر این مساله که در آینده به بهترین شـکل کارها را انجام دهید متمرکز شوید. به طور کلی انتقاد امر مهمی اسـت و بهتر اسـت آن‌چه را که شـنیده‌اید مـورد تجزیه و تحلیـل و ارزیابی قرار دهید، مشـخص کنید که آیا این یک انتقاد معتبر و صحیح اسـت یا خیر و تصمیم بگیرید که چـکار می توانیـد بکنید که مشـکل را حل کنیـد یا اشـتباه را اصـلاح کنیـد. [اگر این موضوع شـکایتی است که شـما مرتبا آن را می شنوید باید فکر کنید که شما از شرایط و وضعیت به وجود آمده چه چیزی می توانید یاد بگیرید که این کار دوباره رخ ندهد.]

جیـم وایت‌هارسـت رو بـه مخاطبیـن در کنفرانس مجازی متذکر شد که تبدیل شدن به یک شـرکت متن‌باز دیگـر یک مزیت به شمار نمی‌رود بلکه به بقای شرکت‌ها مرتبط گشته است. مادامی که شرکت‌های بیشتری در حال خدمت‌محور شـدن هسـتند، بهتر است، شرکت‌های دیگر نیز به چنین مواردی اهمیـت دهنـد تا بتواننـد بـه موفقیت‌های بیشتری دست‌یابند.

در آخر این‌جاست که وصله نرم‌افزاری برای یک پردازشـگر کلمه یـا مدیریت یک کتاب برای یک رهبر یکسـان خواهـد بود. تمامی این‌هـا به این به‌روزرسـانی نیـاز دارند؛ یکی به‌روزرسـانی نرم‌افـزار و دیگری به‌روز کردن بـرای بهبود قابلیت رهبری و کار جمعی در یک جمع و به‌صورت گروهی.■

مباحث ساینده و مکالمات‌هستند که باعث اتخاذ تصمیم‌درست می‌گردند. وی می‌افزاید که شما به پوستی کلفت هم احتیاج دارید؛ تا به نظرات و ایده‌های آنان توجه کنید

اپل را نیز، از متن‌باز شدن گریزی نیست

متن‌باز شدن دیگر اختیاری نیست حتی برای اپل

نویسنده: کیانوش فینی

مهمترین نکته کنفرانس توسعه‌دهندگان اپل در سراسر جهان تنها مختص به معرفی نسخه‌های جدیدی از اواس-ایکس و آی-اواس یا حتی سرویس جدید اپل با نام اپل موزیک نبود بلکه زمانی بود که معاون اپل مهندسی کریگ فدریگی به روی صحنه آمد و اعلام کرد که سوئیفت «Swift» زبان برنامه‌نویسی اختصاصی اپل متن‌باز گشته است. نکته هیجان‌انگیز ماجرا این است که اکثر توسعه‌دهندگان در حال حاضر به سمت توسعه محصولات با بهره‌گیری از ابزارهای آزاد / متن‌باز سوق یافته و استفاده از این ابزارها طی پنج سال گذشته افزایش قابل توجهی داشته است. همچنین اپل کاربران را به این سو سوق داده است که بهتر است از برنامه‌های خانگی اپل استفاده کنند و ابزار شخص ثالثی چون ادوبی فلش دوری گزینند چون معتقد است ابزاری بی‌منفعت هستند. اما به نظر نمی‌رسد حتی اپل هم بتواند جلوی گزینه‌های شخص ثالث بایستد؛ به این دلیل که ابزارهای شخص ثالث بسیاری در دسترس هستند که برای انجام امور توسعه‌ای در اپل به کار بیایند پس اپل برای جلوگیری از استفاده از دیگر ابزارها باید ابزار خود را متن‌باز می‌کرد.

با متن‌باز شدن سوئیفت توسط اپل، این شرکت به توسعه‌دهندگان در دیگر سیستم‌عامل‌ها این امکان را نیز داده است تا به ایجاد برنامه‌های خود برای سیستم‌عامل دیگر نیز با استفاده از کدهای سوئیفت بپردازند؛ که منجر به ایجاد ویرایش مشابه از برنامه‌های کاربردی اپل در سیستم‌عامل های نوپایی مثل اوبونتو تاچ نیز خواهد بود به این شکل که می‌توان کاری کرد که برنامه‌های کاربردی آی-او-اس و نوشته شده در سوئیفت را در اوبونتو تاچ اجرا کرد و یا هر سیستم‌عامل دیگر.

به طور قطع سوئیفت در حال رشد است اما زبان‌های برنامه‌نویسی مشابه دیگری نیز وجود دارند که در چند سال اخیر توسعه داده شده‌اند. به عنوان مثال فیس‌بوک با زبان‌های D و Hack در حال خطا و آزمون است و گوگل نیز زبان جدید Go را در آستین دارد.

در این بین بنیاد موزیلا نیز بیکار نبوده و زبان Rust را توسعه می‌دهد. هر یک از این زبان‌های برنامه‌نویسی دارای قوت و ضعف مختص به خود هستند. هرچه باشد آینده در اختیار چنین زبان‌هایی است که به طور ایده‌آل اکثر آنان متن‌باز هستند. از گزینه اپل گرفته تا گزینه موزیلا.

مایکروسافت نیز سال گذشته اعلام کرد که چارچوب دات‌نت را متن‌باز خواهد کرد و با نام «ویژوال‌استودیو کد»، قصد گسترش نفوذ ابزارهای توسعه نرم‌افزار خود به دیگر سیستم‌عامل‌ها دارد. مایکروسافت همچنین تلاش زیادی برای استفاده کاربران و توسعه‌دهندگان دیگر سیستم‌عامل‌ها از محصولات توسعه‌ای و زبان‌های خود دارد و در حال حاضر در حال تلاش برای فراهم کردن امکانی برای نصب برنامه‌های نوشته شده توسط دات‌نت در سیستم‌عامل‌های آی-او-اس و او-اس-ایکس است. به طوری که حتی بتوان برنامه‌های آی‌اواس را در ویندوز و یا اواس‌ایکس و با استفاده از ویژوال استودیو نوشته و به راحتی در آی-او-اس اجرا کنند. بنابراین با چنین شرایطی اپل چاره‌ای نداشت جز این که زبان برنامه‌نویسی جدید خود را به یک زبان توسعه‌پسند، تبدیل کند.

با این حال اپل در استفاده از متن‌باز قریب نیست. آن‌ها از داروین که انشعابی از بی‌اس‌دی است بهره گرفته و سیستم‌عامل او-اس-ایکس را به یک سیستم‌عامل شبه-یونیکس تبدیل کردند در ضمن وب‌کیت را نیز با استفاده از توسعه پروژه متن‌باز کی‌اچ‌تی‌ام‌ال ایجاد کردند که موتور اصلی برنامه کانکرور بود که با تبدیل به وب‌کیت به موتور اصلی سافاری و بعدها کروم مبدل گشت. همچنین «سامانه عمومی چاپ در یونیکس» CUPS هم ابزاری است متن‌باز که توسط اپل توسعه داده‌می‌شود و در سطح گسترده‌ای در توزیع‌های گنو/لینوکس مورد استفاده است.

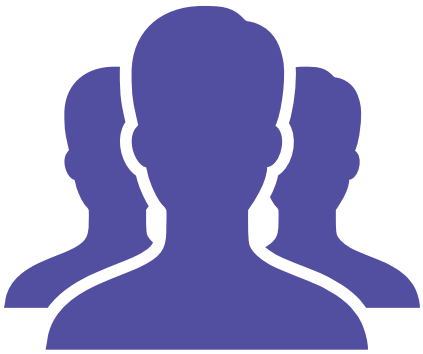
در هر حال اکثر توسعه‌دهندگان اپل از سوئیفت و Objective C استفاده خواهند کرد ولی اپل می‌خواهد تا با متن‌باز کردن سوئیفت سر مایکروسافت را در زمینه نوشتن برنامه‌های بومی سیستمش ببرد و با به جان خریدن ریسک استفاده از زبان بومی خود، در رقابتی چون اندروید به تقویت زبان خود بپردازد حتی اگر در سیستم‌عامل دیگری نیز استفاده شود.

تا چه حد متن‌باز است؟

اپل قرار است هسته اصلی این زبان و تمامی موارد مرتبط هم چون کامپایلر را تحت مجوز خاص خودش متن‌باز نماید که نشان‌دهنده این است که تا چه مقدار متن‌باز خواهد بود. به نظر می‌رسد هم چون فیس‌تایم که تا مدتی متن‌باز شد، سوئیفت هم از حالت متن‌باز خارج شود یا همانند جاوا به صورتی محدود باشد. با این حال استفاده از این زبان در سیستم‌عامل دیگر، برای سوئیفت یک تغییر بزرگ خواهد بود.

اپل بدون نیت، خود به اجرای برنامه‌های سوئیفت در گنو/لینوکس کمک کرده است، با متن‌باز شدن زبان سوئیفت کاربران و توسعه‌دهندگان لینوکس به راحتی می‌توانند از نحوه کار آن آشنا شوند و یا حتی با استفاده از آن زبان‌های دیگر مانند Rust موزیلا را ارتقاء دهند. با این وجود برای اجرای برنامه‌های آی-آواس و اواس-ایکس در گنو/لینوکس نیاز به متن‌باز شدن کوکو (رابط برنامه‌نویسی) نیز هست. کوکو (Cocoa) رابط برنامه‌نویسی شیءگرای محلی اپل برای سیستم‌عامل اواس ده است.. این کار لازم است؛ تا کاربران بتوانند به اجرای برنامه‌های آی-آواس بپردازند. اپل در این مورد حرفی نگفته و یا نگارنده چیزی نشنیده است.

در هر صورت در استفاده از سوئیفت نیز نباید بسیار خوش‌بین بود! با این حال متن‌باز شدن زبان یا نرم‌افزارهایی این چنین که شرکت‌های بزرگ پشت آنان هستند برای مطالعه کدمنبع آنان و یا استفاده از روش‌ها و الگوریتم‌های مورد استفاده آنان و بهره‌گیری درست از این روش‌ها باعث بهبود دیگر گزینه‌ها نیز خواهد شد. هرچند که برنامه‌نویسی این روزها بدون دیدن اخطاری از جانب یک وکیل کاملا کم پیش‌می‌آید■



جامعه کاربری

تدابیری که دولت‌ها می‌توانند برای ترویج نرم‌افزار آزاد به کار گیرند

مترجم:
مهدی بیگی

این مقاله سیاست‌هایی را برای تاثیرگذاری عمیق و پایدار تلاش‌هایی که برای ترویج نرم‌افزار آزاد در سطح دولت و به طبع آن هدایت کشور به سمت نرم‌افزار آزاد انجام می‌شود پیشنهاد می‌دهد.

هدف یک دولت ساماندهی جامعه برای فراهم کردن آزادی و رفاه مردم است. یکی از جنبه‌های این ماموریت در زمینه محاسبات کامپیوتری، تشویق کاربران برای پذیرفتن نرم‌افزار آزاد است: نرم‌افزاری که آزادی کاربران احترام می‌گذارد.

یک نرم‌افزار انحصاری (غیرآزاد) آزادی کاربرانی را که از آن استفاده می‌کنند پایمال می‌کند. این یک مشکل اجتماعی است که دولت باید برای ریشه‌کن کردنش تلاش کند.دولت باید برای حفظ حاکمیت خود بر محاسباتش بر استفاده از نرم‌افزار آزاد اصرار کند. تمام کاربران مستحق این هستند که بر محاسباتشان کنترل داشته باشند اما دولت در مقابل مردم مسئول است که کنترل خودش را بر روی محاسباتی که از طرف مردمش (و با داده‌های آن‌ها) انجام می‌دهد حفظ کند. بخش زیادی از فعالیت‌های دولتی اکنون بر محاسبات کامپیوتری استوار است و کنترل این فعالیت‌ها در گرو کنترل کردن محاسبات صورت گرفته است. از دست دادن این کنترل در اداره‌ای که ماموریت‌های حیاتی دارد می‌تواند امنیت ملی را به مخاطره بیاندازد.

مهاجرت ادارات دولتی به نرم‌افزار آزاد فواید ثانویه‌ای را نیز به دنبال دارد، مانند صرفه جویی در هزینه‌ها و تشویق و توسعه کسب و کارهای محلی مبتنی بر پشتیبانی نرم‌افزار.

در این متن «نهادهای دولتی» به تمامی بخش‌های دولتی مانند بخش‌های عمومی شامل مدرسه‌ها، نهادهای مشارکتی عمومی و خصوصی، فعالیت‌هایی که دولت پشتیبانی می‌کند مانند مدرسه‌های خصوصی و شرکت‌های خصوصی‌ای که توسط دولت کنترل می‌شوند یا با اجازه ویژه یا دستورات دولتی کنترل می‌شوند، اشاره می‌کند.

آموزش

مهم‌ترین سیاست در این میان مربوط به بخش آموزش است چرا که آینده کشور را به راحتی شکل می‌دهد:

فقط نرم‌افزار آزاد آموزش داده شود:

تمام فعالیت‌های آموزشی یا حداقل آن‌هایی که توسط نهادهای دولتی ارائه می‌شوند باید تنها نرم‌افزار آزاد را تدریس کنند (بنابراین آن‌ها هرگز نباید دانش آموزان را به سمت استفاده از یک نرم‌افزار غیرآزاد ترغیب کنند) همچنین باید دلایل مدنی‌ای که باعث اصرار بر استفاده از نرم‌افزار آزاد می‌شود را به دانش آموزان آموزش داد چرا که آموزش نرم‌افزار غیرآزاد آموزش وابستگی است که این در تضاد با اهداف یک مدرسه است.

دولت و مردم

همچنین سیاست‌های دولت در مورد اعمال نفوذ بر نرم‌افزارهایی که شرکت‌ها و سایر افراد استفاده میکنند نیز بسیار حیاتی است:

هرگز اجباری بر استفاده از نرم‌افزار انحصاری وجود نداشته باشد:

قوانین و شیوه‌های اجرای بخش دولتی باید تغییر کند به نحوی که آن‌ها هرگز سازمان‌ها و افراد مستقل را مجبور به استفاده از نرم‌افزار غیرآزاد نکنند یا بر آن‌ها فشار وارد نکنند. همچنین آنان باید جوامع و شیوه‌های عمومی‌ای که چنین پیامدی را به دنبال دارند دلسرد کند (مانند DRM).

تنها نرم‌افزارهای آزاد توزیع شوند:

هنگامی که یک نهاد دولتی نرم‌افزاری را برای عموم منتشر می‌کند مانند نرم‌افزارهایی که در صفحات وب‌شان ارائه یا مشخص می‌شود این نرم‌افزارها باید به عنوان نرم‌افزار آزاد توزیع شوند و باید توانایی اجرا در بسترهای تماما مبتنی بر نرم‌افزار آزاد را داشته باشند.

وب‌سایت‌های دولتی

وب‌سایت‌های نهادهای دولتی و خدمات مبتنی بر شبکه باید به گونه‌ای طراحی شوند که کاربران بدون لطمه خوردن بتوانند از آن‌ها استفاده کنند.

تماما با استفاده از نرم‌افزار آزاد.

قالب‌ها و پروتکل‌های آزاد

نهادهای دولتی تنها باید از قالب‌ها و پروتکل‌های ارتباطی‌ای استفاده کنند که به خوبی توسط نرم‌افزارهای آزاد پشتیبانی شوند ترجیحا با مستندات و قواعد منتشر شده. (منظور ما از این قسمت «استانداردها» نیستند زیرا این قواعد باید در مورد اجزای منطبق با استاندارد و نیز اجزای غیر منطبق اعمال شوند)

باز کردن بند مجوزها از پای کامپیوترها

فروش کامپیوترها نباید نیازمند خرید یک نرم‌افزار انحصاری باشد. فروشنده باید از طرف قانون ملزم به پیشنهاد گزینه خرید کامپیوتر بدون نرم‌افزار انحصاری و بدون پرداخت هزینه مجوز به خریدار باشد.

تحمیل هزینه‌ها اشتباه دیگری است که در این میان رخ می‌دهد و نباید توجه ما را از بی‌عدالتی ذاتی موجود در نرم‌افزارهای انحصاری منحرف سازد. از دست دادن آزادی‌ای که نتیجه استفاده از آن نرم‌افزارهاست. با این حال سواستفاده از اجبار کاربران برای پرداخت، برتری ناعادلانه دیگری را شامل حال توسعه‌دهندگان نرم‌افزارهای انحصاری می‌کند که برای آزادی کاربران مضر است. برای یک دولت شایسته است که جلوی چنین سواستفاده‌ای را بگیرد.

حاکمیت محاسباتی

برخی سیاست‌ها، حاکمیت محاسباتی دولت را به مخاطره می‌اندازند. نهادهای دولتی باید کنترلشان را بر محاسباتی که انجام می‌دهند حفظ کنند نه این که این کنترل را به بخش خصوصی واگذار نمایند. این اصول در مورد تمام کامپیوترها از جمله تلفن‌های هوشمند نیز صدق می‌کند.

مهاجرت به نرم‌افزار آزاد

نهادهای دولتی باید به نرم‌افزار آزاد مهاجرت کنند و نباید نرم‌افزارهای انحصاری را نصب و یا به استفاده از آن‌ها ادامه دهند مگر به شکل موقتی و زودگر. تنها یک نهاد باید قدرت اعطای مجوز استفاده از نرم‌افزار انحصاری به شکل موقتی را داشته باشد آن هم تنها در شرایطی که دلایل قانع‌کننده برای این کار وجود داشته باشد. هدف این نهاد باید به حداقل رساندن این استثنائات تا رسیدن به مرز صفر باشد.

توسعه راه حل‌های آزاد مربوط به فناوری اطلاعات

وقتی که یک نهاد دولتی برای توسعه یک راه حل محاسباتی هزینه می‌کند قرار داد باید پیمانکار را مجبور به ارائه نتیجه به عنوان نرم‌افزار آزاد نماید همچنین طراحی باید به شکلی باشد که یک شخص ثالث بتواند نرم‌افزار را بر روی یک سکوی صد در صد آزاد اجرا کند و توسعه دهد. این مورد باید در تمام قراردادها الزامی باشد. در نتیجه اگر توسعه‌دهنده موافق این الزامات نبود نتوان برای آن کار هزینه کرد.

انتخاب کامپیوترها برای نرم‌افزار آزاد

وقتی که یک نهاد دولتی اقدام به خرید یا اجاره کامپیوتر می‌کند باید از بین مدل‌هایی که (در کلاس خودشان) نزدیک‌ترین سازگاری را برای کار بدون هیچ گونه نرم‌افزار انحصاری دارند دست به انتخاب بزند. دولت باید برای هر رده از کامپیوترها یک فهرست مجاز با توجه به این معیار تهیه کند. مدل‌هایی که هم برای دولت و هم عموم مردم در دسترس باشند باید به مدل‌هایی که تنها برای دولت در دسترس هستند ترجیح داده شوند.

مذاکره با تولید کنندگان

دولت باید به شکل فعال در تمامی حوزه‌های مربوطه با تولیدکنندگان در مورد ارئه محصولات سخت‌افزاری قابل قبولی که نیازمند اجرای نرم‌افزار انحصاری نباشند در بازار (هم برای بخش دولتی و هم برای عموم مردم) مذاکره کند.

اتحاد با سایر دولت ها

دولت باید از سایر دولت‌ها نیز برای مذاکره به شکل جمعی با تولیدکنندگان در مورد محصولات سخت‌افزاری مورد قبول دعوت به عمل آورد چرا که احتمال موفقیت به شکل جمعی بیشتر است.

حاکمیت محاسباتی II

حاکمیت محاسباتی (و امنیتی) دولت شامل کنترل بر کامپیوترهایی که کارهای دولتی را انجام می‌دهند نیز می‌شود. این حاکمیت شامل پرهیز از استفاده از خدمات نرم‌افزار به عنوان سرویس دهنده (مگر این که سرویس توسط یک نهاد دولتی مربوط به همان شاخه از دولت ارائه شود) و همچنین سایر شیوه‌هایی که از میزان کنترل دولت بر محاسباتش می‌کاهد خواهد شد.

دولت باید کامپیوترهای خودش را کنترل کند

هر کامپیوتری که دولت استفاده می‌کند باید متعلق به همان شاخه از دولت باشد یا توسط همان شاخه اجاره شده باشد و آن شاخه نیز نباید به مجموعه‌های دیگر حق تصمیم‌گیری در مورد این که چه کسی دسترسی فیزیکی به کامپیوتر داشته باشد، چه کسی مسئولیت نگهداری (سخت‌افزاری یا نرم‌افزاری) را داشته باشد یا چه نرم‌افزاری باید روی آن نصب شده باشد را اعطا نماید. اگر کامپیوتر قابل حمل نباشد در آن صورت درهنگام استفاده باید در محلی قرار گرفته باشد که در تملک دولت باشد (به شکل ملکی یا اجاره‌ای)

ترغیب به توسعه

سیاست‌های دولت بر توسعه نرم‌افزارهای آزاد و غیرآزاد تاثیرگذار است :

پشتیبانی از نرم‌افزار آزاد

دولت باید با روش‌هایی توسعه‌دهندگان را برای ایجاد یا بهبود نرم‌افزارهای آزاد و فراهم کردن آن برای عموم تشویق کند. مانند معافیت‌های مالیاتی و سایر انگیزه‌های مالی. به صورت مخالف ، این انگیزه‌ها نباید برای توسعه، توزیع یا استفاده از نرم‌افزارهای انحصاری وجود داشته باشند.

عدم حمایت از نرم‌افزار غیرآزاد

به شکل خاص، توسعه‌دهندگان نرم‌افزارهای انحصاری نباید توانایی اهدای نسخه‌هایی از نرم‌افزارشان را به مدارس و در قبال آن تقاضای تخفیف مالیاتی در قبال ارزش اسمی نرم‌افزارشان را داشته باشند. نرم‌افزارهای انحصاری نباید در مدارس قانونی باشند.

ضایعات الکترونیکی (E-waste)

نرم‌افزارهای قابل جایگزینی

بسیاری از کامپیوترهای مدرن به گونه‌ای طراحی شده‌اند که جایگزین کردن نرم‌افزارهای پیش‌فرض در آنان با نرم‌افزارهای آزاد غیرممکن باشد بنابراین تنها راه آزاد سازی آن‌ها دور انداختن آن‌هاست. این عمل برای جامعه مضر است.

بنابراین این کار باید غیرقانونی باشد یا حداقل به شکل قابل ملاحظه‌ای با استفاده از مالیات‌های سنگین محدود شود. این محدودیت باید بر روی فروش، واردات یا توزیع عمده کامپیوترهای نو (نه دست دوم) یا محصولات کامپیوتری‌ای که در مورد اتصالات سخت‌افزاری شان پنهان کاری می‌کنند یا برای توسعه ، نصب و یا جایگزینی کل سیستم یا هریک از نرم‌افزارهای نصب شده که تولید کننده توانایی به‌روزرسانی آن را داشته باشد محدودیت‌های عمدی ایجاد می‌کنند اعمال شود. به طور خلاصه تمام دستگاه‌هایی که برای نصب یک سیستم عامل متفاوت نیاز به «jailbreaking» دارند یا آن‌هایی که اتصالات مورد استفاده برای برخی لوازم جانبی شان مجهول است.

بی‌طرفی در فناوری

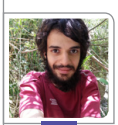
با تدابیر ذکر شده در این مقاله ، دولت می‌تواند کنترل خویش بر محاسباتش را بازگرداند و شهروندان، کسب و کارها و سازمان‌های کشور را به سوی کنترل بر روی محاسباتشان رهنمون سازد. هرچند مخالفت‌هایی وجود دارد که این امر اصل بی‌طرفی در مورد فناوری را نقض می‌کند.

ایده بی‌طرفی در فناوری این است که دولت نباید به شکل دلخواه تمایلات خودش را در موارد تکنیکی اعمال کند. هر چند درستی این اصل اعتراض‌پذیر است اما این اصل به مواردی محدود است که تنها مشکل موجود در آن‌ها مشکل فنی است. تدابیری که در این مقاله از آن‌ها دفاع شده در مورد مسائل اخلاقی، اجتماعی و خطرات سیاسی هستند. در نتیجه این رهنمودها خارج از چهارچوب بی‌طرفی در فناوری هستند.

تنها کسانی که خواستار درهم شکستن یک کشور هستند می‌توانند پیشنهاد دهند که دولتشان در مورد حاکمیتش یا آزادی شهروندانش بی‌طرف باشد.■

نقش جامعه و دولت در حمایت از نرم‌افزار آزاد

نویسنده: سعید علیجانی



نویسنده: مسعود صدرزاد

در این نوشتار سعی شده ابتدا به صورت اجمالی وضع کنونی، اهداف و دستاوردهای دولت در استفاده، ترویج و توسعه نرم‌افزار آزاد در ایران تحلیل شود. در «شناخت جایگاه دولت و جامعه در حمایت از نرم‌افزار آزاد» این اهداف با واقعیت مقایسه شده و در نهایت «راهکار پیشنهادی» نوشته شده است.

اهداف و دستاوردهای دولت

اگر دستاوردهای سازمان فناوری اطلاعات را در راستای اهداف دولت در نظر بگیریم می‌توان این اهداف را در سه دسته کلی تقسیم‌بندی کرد. دسته اول استفاده از نرم‌افزار آزاد است. حتی از واژه به‌کارگیری در نام این تشکیلات استفاده شده است، که نشان‌دهنده اهمیت این هدف برای سازمان فناوری اطلاعات است. دسته دوم ترویج استفاده از نرم‌افزار آزاد است به این معنی که دیگران هم از نرم‌افزار آزاد استفاده کنند و دسته سوم توسعه نرم‌افزار آزاد است. پیش از این‌که به بررسی خود این اهداف بپردازیم برای هر کدام یک مثال بارز خواهیم آورد.

استفاده از سیستم‌عامل اوبونتو روی کامپیوترهای دسکتاپ کارکنان سازمان فناوری اطلاعات یک گام رو به جلو در راستای استفاده از نرم‌افزار آزاد بوده است. افزایش آگاهی جامعه از طریق برگزاری همایش‌ها یا حمایت از گروه‌هایی نظیر لاگ تهران می‌تواند در راستای ترویج استفاده از نرم‌افزار آزاد باشد. حمایت از سیستم‌عامل بومی زمین هم می‌تواند در راستای توسعه نرم‌افزار آزاد شناخته شود.

جهانی‌سازی

نرم‌افزار آزاد بخشی از فرایند جهانی‌سازیست. جهانی‌سازی به معنای متصل کردن تمام مردم دنیا به یکدیگر و حذف مرزهای جغرافیایی. یعنی فراهم کردن شرایطی که مردم دنیا فارغ از زبان، ملیت، مذهب و... بتوانند با یکدیگر تعامل داشته باشند و درکنار هم و با هم زندگی کنند. کشورهای توسعه‌یافته که اغلب شعار جهانی‌سازی می‌دهند، برخلاف ادعای خود عمل کرده‌اند چرا که روزبه‌روز سفر به کشورهای توسعه‌یافته سخت‌تر می‌شود و مرزها بین کشورها پررنگ‌تر می‌شود اما نرم‌افزار آزاد بدون آن‌که شعار جهانی‌سازی را داده باشد، به کمک اینترنت مردم تمام دنیا را به هم متصل کرده است. بالاترین شکل تعامل، تعامل برای رسیدن به یک هدف مشترک است و این در نرم‌افزار آزاد نمود پیدا کرده است. زیرا جامعه نرم‌افزار آزاد خلق یک نرم‌افزار را برای رفع یک نیاز مشترک، هدف قرار داده است و امروزه نمی‌توان روی نرم‌افزارهای آزاد برچسب کشور یا شرکت به‌خصوصی را زد.

جامعه نرم‌افزار آزاد خلق یک نرم‌افزار را برای رفع یک نیاز مشترک، هدف قرار داده است و امروزه نمی‌توان روی نرم‌افزارهای آزاد برچسب کشور یا شرکت به‌خصوصی را زد.

شناخت جایگاه جامعه و دولت در حمایت از نرم‌افزار آزاد

دولت‌ها، جامعه و شرکت‌ها انگیزه‌های متفاوتی برای حمایت از نرم‌افزار آزاد دارند چرا که نیاز آن‌ها برای استفاده از نرم‌افزار آزاد متفاوت است. اگر نیاز هر کدام به درستی تشخیص داده شود می‌توان درباره نقاط اشتراک آن‌ها بحث کرد و این اهداف را در راستای هم قرار داد. هدف دولت‌ها از استفاده از نرم‌افزار آزاد معمولا به امنیت بیشتر اطلاعاتشان مربوط می‌شود. شرکت‌ها به

دنبال راه‌حل‌های کم‌هزینه‌تر و قابل اعتمادتر برای حل مسائل خود هستند و جامعه به دنبال راهکاریست که نیازهای خود را برطرف کند.

بخش قابل توجهی از قدرت و ثروت به ترتیب در اختیار دولت‌ها و شرکت‌هاست، در حالی که توسعه نرم‌افزار آزاد نیازی به قدرت و ثروت ندارد. به این معنی نیست که این دو نمی‌توانند به توسعه نرم‌افزار آزاد کمک کنند بلکه به این معنیست که نرم‌افزار آزاد بدون قدرت دولت‌ها و ثروت شرکت‌ها هم می‌تواند توسعه یابد همان طور که بدون این دو به جایگاه کنونی خود رسیده است.

اگر دولت متوجه سهم کم قدرت در توسعه نرم‌افزار آزاد باشد، جایگاه خود را تشخیص می‌دهد و به اندازه سهم خود و در جای مناسب، مسئولیت اجتماعی خود را ایفا می‌کند. اگر چنین شود، دولت متوجه می‌شود که مجبور کردن افراد یک سازمان دولتی به نصب اوبونتو روی کامپیوترهای شخصی خود، کمکی به توسعه نرم‌افزار آزاد نمی‌کند، به دلیل این که استفاده افراد به عنوان کاربر نهایی از یک سیستم‌عامل آزاد نقش چندان موثری در توسعه نرم‌افزار آزاد نخواهد داشت. همچنین ایجاد یا حمایت از ایجاد سیستم‌عامل زمین به دلیل عدم استفاده از این سیستم‌عامل توسط شرکت‌های مبتنی بر ابر، چه در داخل و چه در خارج کشور کمکی به توسعه نرم‌افزار آزاد نمی‌کند نه به این خاطر که لزوما این سیستم‌عامل ضعیف عمل کرده یا کم هزینه شده، بلکه به این دلیل که این سیستم‌عامل از ابتدا هم قرار نبوده توسط مردم جهان مورد استفاده قرار بگیرد و یک سیستم‌عامل ملی/بومی بوده. این در حالیست که یکی از افتخارات دولت استفاده از سیستم‌عامل غیربومی اوبونتو است که بخش قابل توجهی از توسعه آن توسط جامعه کاربری آن از سراسر جهان صورت می‌گیرد. ابهامی که در واژه ملی قرار دارد آیا ناشی از عدم شناخت ماهیت نرم‌افزار آزاد و ارتباط آن با فرایند جهانی‌سازی است؟ آیا اجباری در کار است که دستگاه‌های دولتی از این اسامی برای دریافت تایید نام سازمان و پروژه‌های آزاد استفاده می‌کنند؟

راهکار پیشنهادی

در بخش قبل گفتیم نرم‌افزار آزاد به قدرت و ثروت نیاز ندارد. نرم‌افزار آزاد به جای این دو به جامعه مشارکت‌کننده نیاز دارد. نرم‌افزارهای آزاد در سراسر دنیا توسط جوامع کاربری تولید، آزمایش و منتشر می‌شود و شرکت‌ها و دولت‌ها تنها حامی این جوامع و گروه‌های کاربری هستند. پرسشی که در این‌جا مطرح می‌شود اینست که پس دولت چه نقشی می‌تواند در توسعه نرم‌افزار آزاد داشته باشد؟ به این پرسش در دو بخش جداگانه پاسخ می‌دهیم. بخش اول شناسایی نیازهای جوامع و رفع بخشی از آن که در توان دولت می‌باشد و بخش دوم تالیف سیاست‌های درست است.

شناسایی نیازهای جوامع و رفع بخشی از آن‌ها

برای مورد دوم به یک دایرکتوری از پروژه‌های نرم‌افزار آزاد و دانش آزاد از سراسر دنیا نیاز هست برای همین ابتدا باید معیارهایی مشخص شود که بر اساس آن هم آزاد بودن پروژه‌ها و هم صلاحیت آن‌ها چه از نظر فنی و چه از نظر توان اجرایی مورد بررسی و تایید قرار بگیرد. بعد از تهیه دایرکتوری گام بعد مشخص کردن نیازهای هر پروژه و حیطه تخصصی هر کدام هست. برای مثال سایت گنو، لانچپد، داک‌داک‌گو، ویکیپدیا و.. امکان ترجمه به صورت گروهی را فراهم کرده‌اند و هر فردی که به زبان انگلیسی مسلط باشد، می‌تواند به این پروژه‌ها کمک کند. ممکن است بعضی از پروژه‌ها نیازمند گرافیست، بعضی ایجاد مستندات فنی، بعضی برنامه‌نویس یا حتی حمایت مالی (سیستم دونیشن را هم شامل می‌شود) و.. داشته باشند. بعد از طبقه‌بندی نیازها باید برای هر پروژه و تخصص راهنمایی تهیه کرد که شامل روال مشارکت در آن پروژه باشد. البته تمام نیازها در این طبقه‌بندی قرار نمی‌گیرد. برخی نیازها ممکن است حالت خاص‌تری داشته باشد ولی نکته مهمی که وجود دارد اینست که نیاز را افرادی تشخیص می‌دهند که در این پروژه‌ها کار می‌کنند نه نگارندگان این مقاله یا کارکنان سازمان‌های دولتی.

جمع‌بندی مراحل پیشنهادی برای ایجاد بستر شناسایی نیازهای جوامع و حمایت از آن‌ها بدین است:

- صفر- مشخص کردن معیارها و حداقل‌های مورد نیاز برای آزاد بودن یک پروژه و صلاحیت فنی و اجرایی آن
- یک- تهیه دایرکتوری جوامع و پروژه‌ها و معرفی آن‌ها
- دو- دسته‌بندی جوامع فعال در حوزه نرم‌افزار یا دانش آزاد
- سه- بررسی نیازها از طریق پرس‌وجو از فعالین آن حوزه
- چهار- طبقه‌بندی نیازها بر حسب تخصص‌ها یا منابع مالی مورد نیاز

سیاست‌ها

هرچند مهاجرت دولت و توابع آن از جمله مدارس و سازمان‌ها و شرکت‌های دولتی به نرم‌افزار آزاد می‌تواند فوایدی از جمله ذخیره شدن منابع مالی و تشویق و گسترش شرکت‌های پشتیبان نرم‌افزار در داخل کشور، اثر نکردن فشارهای دولت‌ها برای تحریم ایران توسط شرکت‌های انحصاری و... داشته باشد اما ماموریت اصلی دولت سازماندهی جامعه برای آزادی و سعادت مردم است. یکی از جنبه‌های این آرمان در حیطه کامپیوتر تشویق کاربران به فراگیری نرم‌افزار آزاد یعنی نرم‌افزارهایی که به آزادی کاربران احترام می‌گذارند، است در حالی که نرم‌افزار انحصاری آزادی کاربرانش را پایمال می‌کند و این یک مشکل اجتماعیست که دولت باید آن را ریشه‌کن کند.

مجبور کردن افراد یک سازمان دولتی به نصب اوبونتو روی کامپیوترهای شخصی خود، کمکی به توسعه نرم‌افزار آزاد نمی‌کند، به دلیل این که استفاده افراد به عنوان کاربر نهایی از یک‌سیستم‌عامل آزاد نقش چندان موثری در توسعه نرم‌افزار آزاد نخواهد داشت

مهمترین سیاست در حوزه آموزش مطرح است زیرا آموزش آینده یک کشور را می‌سازد. فعالیت‌های آموزشی و افراد مرتبط با امور آموزشی باید فقط نرم‌افزارهای آزاد را تدریس کنند و نباید هیچوقت دانش‌آموزان را به استفاده از نرم‌افزارهای انحصاری عادت دهند. این افراد باید دلایل اجتماعی پافشاری روی نرم‌افزار آزاد را آموزش دهند چرا که آموزش نرم‌افزار انحصاری آموزش وابستگیست که این با ماموریت مدارس در تضاد است.

جستارهای وابسته

آخرین نسخه مقاله به همراه اسلایدهای این مقاله، ارائه‌شده در نشست هم‌اندیشی توسعه بکارگیری نرم‌افزار آزاد/متن‌باز در کشور در ویکی بنیاد دانش آزاد در دسترس علاقه‌مندان قرار گرفته است.■

گفتگو با بابک فرخی توسعه‌دهنده سولاریس و BSD و مرور فناوری در دهه هفتاد

زندگی با شیطانک قرمز

به هر شرکتی که پا می‌گذاشت دیگر حتی لازم نبود توضیحی بدهد. کافی بود بگوید تمام سیستم‌های سازمان را به BSD تغییر دهید تا بدون هیچ شرط و شروطی آن را بپذیرند. پشتوانه سال‌ها کار تخصصی در پروژه‌های داخلی و ارتباط با چهره‌های بین‌المللی او را در حوزه کاری خودش آوازه خاص و عام کرده بود. به قول خودش: «همیشه برای همه مرموز و عجیب بودم».

نویسنده:
امیر حسین حسینی‌پژوه

یک اتاق کوچک خنک و پر نور و درخشان در شرکت ایمن پردیس خانه اول بابک فرخی است. ترجیح می‌دهد به جای سر و کله زدن با آدم‌ها یا وقت گذراندن در شبکه‌های اجتماعی در این اتاق کوچک پشت آی‌مک ۲۷ اینچی‌اش بنشیند و آخرین پیش‌نویس‌های IETF را مطالعه کند و نظرش را درباره استاندارد «مانیتورینگ کیفیت شبکه‌های موبایل در ابعاد بزرگ» بنویسد یا در کنار کارمند روسی nginx و کارمند آمریکایی netflix در بهبود عمل‌کرد الگوریتم‌های IP forwarding به «پیرمردهای بی‌اس‌دی» کمک کند. تا این حد که همسرش می‌گوید:«گاهی اوقات بچه‌ها هم می‌ترسند به اتاقش بروند که مبادا آن خلوت را به هم بزنند». کنار دستش یک لپ‌تاپ سونی نیز قرار دارد که هنوز منتظر آپدیت ویندوز ده‌اش است. همان لحظه با خوشحالی در حال توسعه سیستم‌عامل شرکت نیز هست و وقتی تمام این کارهای بزرگ را نه مانند یک کار دشوار بلکه مانند یک سرگرمی نشاط‌آور شرح می‌دهد. کتاب‌هایی زیادی سمت راست میزش روی هم چیده شده از فرهنگ سامورایی تا جامعه شناسی و تاریخ باستا و فلسفه و رمان را می‌توان در میان آن‌ها مشاهده کرد. یک کیندل در شرکت دارد و یکی در خانه و می‌گوید: «برای من وظیفه است که هرشب قبل از خواب کتاب بخوانم»

به نظر بابک فرخی لینوکس برای یک محیط «کاتینگ اج» بوده و برای توسعه و پیشروی مناسب‌تر است . او هرگز در جمع‌های لینوکسی پا نگذاشته و خودش می‌گوید اجتماعی نیست. روی

هیـچ نرم‌افزاری تعصب ندارد. جمله‌ای که بارها برای ابزارهـای مختلـف تکرار می‌کند این اسـت که «من هم خیلی باهاش خوشحالم» و این تنها دلیلش برای کار کردن با نرم‌افزار یا سـخت افزاری اسـت. به قول خودش ترجیح می‌دهد به جای آدم‌ها با ماشین‌ها سر و کله بزند «۳۶ سـاعت می‌شینم پای کامپیوترم که یک مشکلی را حل کنم و واقعا هم خسته نمی‌شم و هرچه جلوتر می‌روم موتورم گرم‌تر می‌شود ولی حرف زدن با آدم‌ها و مذاکره و فروش اذیتم می‌کنم ترجیح می‌دهم در را ببندم و یک مشکل جهانی را حل کنم».

آپادانا

اوایل دهه ۷۰ اسـت و بابک سـال‌های دبیرسـتان را سـپری می‌کنـد. روزهایی که رایانـه محصولی نو در خانه‌های ایرانی به حسـاب می‌آیـد و به راحتی قابل خریـد نیسـت. اولیـن رایانه عمرش کـه یک آمیگا ۵۰۰ اسـت را در خانـه یکی از دوسـتانش می‌یابد و هرگز فکرش را نمی‌کرد ۲۴ سال بعد فرزندان همین دستگاه بهترین همدمش شوند. آن زمان وقتی کسی رایانـه‌ای را می‌خرید نه خبری از فیلم‌های با کیفیت بـود نـه اینترنتی بنابراین بیشـتر از آن که یک رایانه جنبه مصرفی داشـته باشـد دسـتگاهی بود که افراد کمـی از آن برای سـاخت برنامه اسـتفاده می‌کردند. می‌گوید: «همیشه دوست داشتم یک چیزی بسازم» برای همین سـعی می‌کند یکی از زبان‌های رایانه‌ای را یاد بگیرد و اتفاقی با زبان اسـمبلی موتورولا آشـنا می‌شـود. می‌گوید: «از شانس بد من اولین زبانی که یاد گرفتم اسـمبلی بود، زبان سـختی است اما برای مـن هیجان انگیز بود». برنامه‌نویسـی با ایـن زبان را شروع می‌کند و اتفاقا نرم‌افزارهای جالب و خوبی هم می‌نویسد.

مدتی بعد اولین رایانه شـخصی را می‌بیند. یک PC XT ۸۰۸۸ اسـت. می‌گویـد: «مال یکـی از اقوام بود رفتیم در خیابان جمهوری یک شـرکتی که دو سـه نفـر از آمریـکا آمده بودنـد و رایانه‌های ۲۸۶ دسـت دوم می‌فروختنـد بـا مانیتـور EGA کـه رزولوشـنش ۳۲۰ در ۲۰۰ بـود. الان فکـر کنم یک اپل واچ از این بیشتر باشد». در اولین کار می‌خواهد برای آن هم با اسمبلی برنامه بنویسد اما متوجه می‌شود معماری آن بـا موتـورولا فرق می‌کند به هر حال کم‌کم شـروع یادگیری اسـمبلی اینتل می‌کنـد. می‌گوید: «کتابی مال پیتر نورتون بود که من سـمپل‌ها رو می‌نوشـتم بعد توسعه می‌دادم، فایل منیجر، ادیتور و امثال این‌ها را روی سیسـتم عامل داس۵ و داس ۶ می‌نوشـتم». مدتـی بعد یکی از دوسـتانش مسـیر یادگیـری او را تغییـر داده و زبان سـی را بـه او معرفی می‌کند. یک کتاب فارسـی همـدم او در یادگیری این زبان جدید می‌شود. می‌گوید: «دیدم چقدر من با اسمبلی مکافات می‌کشیدم» با turbo C و از آن طریق با نرم‌افزار دیگر همین مجموعه یعنی turbo pascal آشـنا می‌شـود و شروع به یادگیری پاسکال می‌کند. تمام این اتفاقات در طی دو الی سه سال روی می‌دهد و تنها منبعاش هم کتاب‌های این و آن اسـت. می‌گوید: «یک کتاب بنفش رنگ انگلیسی به من داد که پاسکال ۵/۵ را یاد مـی‌داد دیدم چقدر از سـی راحت‌تر اسـت» و مدتی بعد از طریق دوسـتان دبیرسـتانش با بیسـیک آشنا می‌شـود. در حقیقت به قول خودش مسیر یادگیری برنامه‌نویسی را برعکس می‌رود.

در دوران دبیرسـتان یکـی از دوسـتان بابـک که در الکترونیک سر رشته دارد بردی طراحی می‌کند. این برد به پورت پرینتر وصل می‌شود و با آن می‌توان دو رایانـه را از طریـق باند FM به هم وصل کـرد و از این طریق فایل جابجا کرد. دستگاهی که کارکردش برای الان هم عجیب است. «ما اون موقع FM مودولاسیون می‌دانسـتیم و روی پـورت پرینتـر هـم بلـد بودیـم برنامه‌نویسی کنیم برنامه را با کوییک بیسیک نوشتیم که اینترفیسـی که با پـورت پرینتر صحبت می‌کرد بـا اسـمبلی بـود» به همین نحـو سـال‌های انتهایی دبیرستان با خرده فعالیت‌هایی در زمینه برنامه‌نویسی طی می‌شود.

آن روزهـا اینترنتـی در کار نبـود اما شـبکه‌هایی به نـام BBS یـا Bulletin board system وجود داشـتند که رایانه‌ها را از طریق ترمینال به هم متصل می‌کرد. او با مودمش که ۲۴۰۰ بیت بر ثانیه سـرعت داشت به شـبکه بی‌بی‌اس مروا وصل می‌شد. می‌گوید: «وقتی آدم یاد این خاطرات می‌افتد خنده‌دار به نظر می‌آید اما ما با این مودم چه کارا که نکردیم و چه فایل‌ها که از این BBS دانلود نکردیم»

به سرعت پای اینترنت به ایران باز می‌شود اما اینترنت آفلاین. بابک فرخی که دبیرسـتان را تمام کرده یک روز در مجلـه رایانـه بـا سـرویس ایمیل آپادانا آشـنا می‌شـود و بعد از تماس با سـپهر محمدی مدیر این شرکت تک نفره در خیابان پامنار در حوالی بازار تهران ملاقات می‌کند. اولین تصورش از شـرکت را این‌طور می‌گوید: «دیدم یک مرد اسـت بـا دو کامپیوتر و دو میـز» این شـرکت به مـردم آدرس ایمیل می‌دهد و روزی یک بار به شـبکه دایال‌آپ آمریکا وصل شـده و ایمیل‌ها را می‌فرستاد و می‌گرفت و کیلو بایتی ده تا ۱۵ تومن پول دریافت می‌کرد. این آشنایی ناگهانی باعث می‌شود بابک فرخی ادامه تحصیل را رها کرده و به دنبال علاقه‌مندی و رؤیای خود در آپادانا شروع به کار کند. خانواده هم که عشق و علاقه بابک را که می‌بینند تسلیم تصمیم او می‌شود و به او برای ادامه تحصیل اصرار نمی‌کنند.

سال ۷۴ است. شرکت آپادانا سعی می‌کند فعالیت‌های قبلی خود را توسـعه دهد. بابک روی ویـنـدوز NT 3.5 یک برنامه می‌نویسـد کـه ایمیل‌های دریافت شـده شرکت آپادانا در روز را اصطلاحا دیس‌پچ کرده و همه را در قالـب یک فایل در مـی‌آورد آن‌ها این فایل را با FTP می‌گرفتند. آن زمان ویندوز 3.1 شبکه DialUp و استانداردهای TCP/IP را نداشت بلکه با یک برنامه ثالث به نام trumpet این امکان اضافه می‌شـد و کاربران با برنامه pegasus mail ایمیل‌هایشان را می‌نوشتند و به شبکه آپادانا می‌فرستادند. بابک هم به همراه سپهر هر شـب آن‌ها را با telnet روی یک ماشـین یونیکس در آمریـکا با FTP آپلود می‌کردند می‌گوید: «SSH اختراع نشـده بود فایـل را می‌گذاشـتیم با اجرا کـردن یک کامند ایمیل‌ها می‌رفت بعد جوابش که می‌آمد سـر یک ساعت دانلود می‌کردیم». می‌گوید: «حالا بگذریم که چقدر باگ داشت و ایمیل یکی را در میل باکس دیگری می‌گذاشـتیم» و این ۵۰ کاربر می‌دانستند کـه سـاعت شـش بعـد از ظهر بایـد وصل شـوند و ایمیل‌هایشان را بگیرند. می‌گوید: «۸ تا مودم داشتیم با ۸ تا خط تلفن گاهی کسی تاجر بود و زنگ می‌زد که تو رو خدا الان وصل شوید ایمیل مهم دارم».

ایـن دوران به سـرعت بـاد می‌گذرد و پـای اینترنت ماهواره‌ای به کشور باز می‌شود. از شرکت داده‌پردازی یک خط ارسال کننده دایال‌آپ و یک دریافت‌کننده ماهواره‌ای می‌خرند. ایمیل آنلاین می‌شود ولی اینترنت هنوز آفلاین است برای همین با سرویس‌هایی به نام web to mail gateway این مشکل را بر طرف می‌کنند. کافـی بود بـه یک آدرس خـاص URL را ایمیل کنید تـا فایل‌های اچ‌تـی‌ام‌ال را به ایمیل اتچ کـرده و برای شما بفرستد. مدتی بعد که کارشان می‌گیرد دفتر را از بازار به قیطریه منتقل می‌کنند و تعدادشان به ۴ نفر افزایش می‌یابد.

مکعب مرموز

یـک روز یک مکعب کوچک مسـیر زندگی حرفه‌ای بابک را تغییر می‌دهد. یکی از شـرکت‌های همکار با یکی از کامپیوترهایش به مشـکل می‌خـورد و آن را برای آپادانا می‌فرستند تا شاید آن‌ها بتوانند دستگاه را درسـت کنند. مشـکل از این قرار اسـت که فلاپی دیسک در دستگاه گیر کرده و هیچ دکمه‌ای هم روی آن نیست تا آن را با فشردن دکمه خارج کنند. ورود ناگهانی یـک SPARCstation IPX به مجموعه آن‌ها که تشنه آموختن هستند اوضاع را به هم می‌ریزد.

یک هفته با این دستگاه کار می‌کنند تا متوجه مشکل

به وجود آمده می‌شوند. سیستم‌عامل این دستگاه Sun OS ۲/۶ بود و برای خروج فلاپی باید دستور eject را در کامندلاین می‌نوشتند. با این حال طعم یک فناوری جدید و یک دنیای متفاوت از ویندوز برای آن‌ها شیرین است و سعی می‌کنند با زیرکی آن را مدت بیشتری پیش خود نگه دارند. چشمانش برقی می‌زند و با شور می‌گوید: «خیلی هیجان انگیز بود» و واژه «خیلی» را چند ثانیه می‌کشد با این امید که ما هم میزان اشتیاقش در آن روز را درک کنیم.

شروع به یادگرفتن سولاریس می‌کنند و تصمیم می‌گیرند یک وب سرور برای آن بنویسند چون نمونه اصلی چندان راضی کننده نیست. از آن به بعد تا امروز دورادور سولاریس را دنبال کرده و تا قبل از سال ۹۰ هرکس که پروژه‌ای در این زمینه دارد پیش او می‌آید. می‌گوید: «من خیلی به سان علاقه داشتم چون فاندر سان آقای بیل جوی را دوست دارم، بیل جوی یک آدم خیلی باهوش است که Vim و Cshell را ساخت انصافا هم سولاریس یک سیستم‌عامل پدر مادر دار روی سخت‌افزار پدر مادر دار است» رم، سی‌پی‌یو و هارد دستگاه‌های سولاریس هات پلاگ است یعنی می‌توان اجزای رایانه را در حالی که روشن است تغییر داد. این مفهوم را خودش خیلی بهتر توضیح می‌دهد. می‌گوید: «مثلا سان فایر ۲۵۰۰۰ یک چیزی اندازه یخچال ساید بای ساید است که درش باز می‌شود و می‌توانید داخلش شوید مثلا ۲۵۶ سی‌پی‌یو دارد، با کنسول کار می‌کنید که می‌گوید سی‌پی‌یو شماره ۵۳ از کار افتاد روی هر سیستم دیگر وقتی سی‌پی‌یو از کار می‌افتد یعنی سیستم از کار افتاده، اما این‌جا شما سی‌پی‌یو را عوض می‌کنید دوباره روی کنسول می‌نویسد سی‌پی‌یو شماره ۵۳ آنلاین شد. دستگاه‌های سان معروف است که وقتی روشنش می‌کنی ۴۰ دقیقه طول می‌کشد بالا بیاید و دیگر خاموشش نمی‌کنی» درباره مزایای سولاریس می‌گوید: «دستگاه سی‌تی اسکنی trusted solaris وجود ندارد که درش سولاریس نباشد یا تراست سولاریس سیستم‌عاملی است که پنتاگون استفاده می‌کند برای اون قدیمی‌تری که با سولاریس کار کرده لینوکس عجیب است لینوکس یک هیپی به حساب می‌آید وقتی پیشش از مزایای لینوکس صحبت کنید ناراحت می‌شود»

با همکارش سپهر محمدی سعی می‌کنند یونیکس را روی رایانه‌های معمولی PC هم نصب کنند برای همین راهی مغازه‌های تهران می‌شوند و در یک مغازه سی‌دی فروشی در چهارراه ولی‌عصر دست تقدیر آن‌ها را به صورت اتفاقی به لینوکس سوزی می‌رساند. ملیت و زبان آلمانی این توزیع دردسر ساز می‌شود. از روی ناچاری و با علاقه شروع به یادگیری آلمانی می‌کند تا شاید زبان این توزیع را بفهمد و بتواند طعم لینوکس را بچشد. مدتی بعد با پذیرش شکست اما همچنان مشتاق به دنبال توزیع دیگری می‌رود و با Slackware آشنا می‌شود. این نخستین توزیعی بود که توانست در سال ۷۵ به لطف محلی که در آن کار می‌کرد از اینترنت دانلود کند. آن زمان روش نصب یک توزیع به این صورت بود که ابتدا باید یک کرنل یک و نیم مگابایتی را از روی فلاپی بوت می‌کردند تا آن هسته لینوکس، درایور سی‌دی رام دستگاه را لود کند. می‌گوید: «کرنل ۲ تازه آمده بود، برای همین کار چهار پنج روز سر و کله زدیم» لینوکس هیچ کدام از سخت‌افزارهای آن‌ها را نمی‌شناسد و با مصیبت یک کارت شبکه Novel NE2000 از بازار رضا می‌گیرند تا در نهایت به اینترنت وصل می‌شوند. کارت گرافیک هم شناسایی نمی‌شود. با خنده می‌گوید: «هی ما کامپایل می‌کردیم از اون موقع من کامپایل کردن‌ام خوب شد» در همین روزها یکی از دوستانش یک ردهت نسخه ۴ را از آمریکا می‌آورد. می‌گوید: «رابط کاربری‌اش بهتر بود و کمی آدم حسابی‌تر بود هنوز

سی‌دی‌اش را دارم»

با این حال ترجیح می‌دهند سیستم‌های شرکت روی ویندوز NT بمانند و برنامه‌ها را تحت ویژوال استودیو توسعه دهند. می‌گوید: «بعد که به شرکت دیگری رفتم کم کم این اعتماد به نفس را پیدا کردم که ساختار را روی یونیکس بیاورم» هرچند با جاوا هم برنامه‌نویسی می‌کند اما این زبان به هیچ وجه نمی‌تواند او را راضی کند و از محیط گرافیکی AWT به بدی یاد می‌کند و وقتی می‌بینند سی تمام نیازهایشان را پاسخ می‌دهد ترجیح می‌دهند به ترند آن روزها توجه‌ای نکنند. می‌گوید: «جاوا از آن تکنولوژی‌هایی بود که از زمان خودش جلوتر بود و به درد نیاز ما نمی‌خورد».

در آپادانا با CGI یک بولتن برد طراحی می‌کنند و نامش را نمایه می‌گذارند. این سرویس یکی از برگ برنده‌های آپادانا می‌شود، یک سایت داخل شبکه که مردم در آن لینک به اشتراک می‌گذاشتند. دیگر محصولشان یک سرویس چت است که سرورش را با جاوا می‌نویسند و کلاینت ویندوزش را با دلفی. این‌ها ایده‌هایی بود که از شبکه‌های BBS به ارث گرفتند. چند سال بعد ICQ و بعدش YAHOO pager یا همان مسنجر معروف وارد رقابت شد.

تا این که یک روز در گشت و گذارهایش با NetBSD آشنا می‌شود. می‌گوید: «خیلی پایدار تر بود و حداقل من راحت‌تر با آن کار می‌کردم» FreeBSD 3.3 آغاز ماجرای او با خانواده یونیکس است. با این که لینوکس و سولاریس را هم دنبال می‌کرده و حتی در توسعه سولاریس مشارکت کرده اما از آن سال به بعد تا امروز سیستم‌عامل مورد علاقه‌اش برای سرور FreeBSD می‌شود. به واسطه همین پروژه با آدم‌های مختلفی در کشورهای دیگر دوست می‌شود. می‌گوید: «الان هم هرکاری انجام می‌دهم یک ربطی به BSD دارد.»

مهاجرت

مدتی بعد حدود سال ۸۰ شرکت آپادانا را ترک می‌کند و به داتک می‌رود و از همان ابتدا تمام سیستم‌ها را بر پایه یونیکس بنا می‌گذارد. روزی که آپادانا را ترک می‌کند سیستم اینترنت تازه از ۱۴ کیلوبیت به ۲۸ کیلوبیت رسیده است. آن روزها تازه موزیلا مرورگرش را عرضه کرده. می‌گوید: «آن موقع از همه بهتر اینترنت اکسپلورر بود، عالی بود، اکتیو ایکس و تمام فناوری‌ها رو هم ساپورت می‌کرد و موزیلا خیلی درب و داغان بود حتی من آن زمان اولین دستگاه اپل را که دیدم بهترین مرورگرش اینترنت اکسپلورر بود» هرچند زمانه به این نرم‌افزار وفا نکرد و اکنون جزو بدترین مرورگرهای دنیا است.

۴ سال در قامت مدیر فنی در داتک می‌ماند. می‌گوید: «یک پسر جوان کم سن و سال بودم که چون از این چیزها سر در می‌آوردم اسمم شد مدیر فنی وگرنه نه مدیریت بلد بودم نه چیزی». با این حال موفق می‌شود تمام ساختار سیستم‌های رایانه‌ای شرکت داتک را تحت BSD نصب کند . درباره ساختار BSD می‌گوید: «تر و تمیزتر است شما در لینوکس برنامه که نصب می‌کنید تنظیماتش را در /etc می‌ریزد اما بی‌اس‌دی می‌گوید تمام روت مال سیستم‌عامل است برای همین لینوکس هم مثل ویندوز بعد از یک مدت چاق می‌شود و همه چیز قاطی می‌شود بعد از مدتی می‌روید در etc حجم زیادی configuration فایل می‌بینید می‌روید در bin تعداد زیادی فایل باینری می‌بینید اما در بی‌اس‌دی و سولاریس این طوری نیست همه می‌رود در //usr local و هر وقت خواستی این پوشه را کلا پاک می‌کنی سیستم‌عامل نو می‌شود» او درباره مزیت‌های بی‌اس‌دی ادامه می‌دهد: «چیزی به نام port داریم و من نرم‌افزارهایم را از فایل‌های باینری نصب نمی‌کنم از سورس کد هم نصب نمی‌کنم که پیچیدگی‌های وابستگی داشته باشد یک چیزی بین پکیج و سورس است موقع کامپایل تصمیم می‌گیری چه آپشن‌هایی روشن باشد و به چه کتابخانه‌ای لینک شود لینوکس هم مزایای خودش را دارد اما با BSD خیلی خوشحال‌ترم» با این حال منکر برتری‌های لینوکس هم نمی‌شود و معتقد است وقتی بخواهد از پایگاه داده‌های اوراکل استفاده کند هیچ وقت سراغ BSD نمی‌رود.

در نزدیکی سال ۸۱ مخابرات سرویس E1 را شروع می‌کند و اوضاع دایال‌آپ مقداری بهتر می‌شود. در همان سال‌ها داتک با مشارکت شرکتی دیگر، ISP شده و بحث PAP و اینترنت ADSL به اتفاق روز جامعه فناوری اطلاعات ایران تبدیل می‌شود. بابک در نوشتن طرح ADSL هم مشارکت می‌کند اما همان روزها از داتک بیرون می‌آید.

ازدواج

سال ۸۴ به شاتل می‌رود هرچند دو سال بیشتر آن‌جا نمی‌ماند. به قول خودش شیب شاتل خیلی زود برایش ملایم شد و دیگر آن‌جا کار کردن جذابیتی نداشت. این را می‌اندازد تقصیر این که شرکت بیش از آن‌که به مسائل فنی اهمیت دهد، دنبال بیزینس خودش بود. به محض ورودش روی تمام ساختار آن‌جا FreeBSD نصب می‌کند. می‌گوید: «این طوری شد که هرجا که رفتم یک ردپایی به جا گذاشتم» مجبور می‌شود هرآنچه می‌داند را به دیگران هم یاد بدهم تا در صورت ترک شرکت مشکلی پیش نیاید. این یاد دادن‌ها باعث می‌شود تا سال ۸۹ شروع به آموزش بی‌اس‌دی با همکاری BSDCertificationGroup که مقر اصلی‌اش در آمریکا بود بکند. نماینده رسمی این مؤسسه می‌شود. روند هم به این صورت است که امتحان می‌گرفتند و پاسخ‌ها را برای مؤسسه اصلی ارسال می‌کردند. با همین روش اکنون ۳۰ نفر در ایران مدرک رسمی BSD-A دارند و بعد از آمریکا ایران دومین کشور از نظر جمعیت متخصصان BSD است. با این حال مشغله کاری نمی‌گذارد این کار را ادامه دهد.

در همان سال‌هایی که در شاتل است با حنانه حاج سید جوادی آشنا می‌شود و در سال ۸۵ ازدواج می‌کند. این ازدواج بعدها در زندگی کاری‌اش هم تأثیر مثبتی می‌گذارد. از نظر او بابک یک گیک است. می‌گویم زندگی با یک گیک چگونه است؟ سکوت سختی می‌کند. شور و شوق اولیه از صدایش رخت بر می‌بندد و آرام می‌گوید: «اوایلش خیلی سخت بود» بعد از کمی سکوت ادامه می‌دهد: «من خیلی با مهندسین همکار بودم ولی بابک با بقیه متفاوت بود اون هیچ وقت ساعت کاری‌اش تمام نمی‌شود. روز تعطیل و مسافرت برایش تعریف ندارد» می‌خندد: « البته الان خیلی بهتر شده» او ادامه می‌دهد: «با گذر سال‌ها من درک کردم که باید این فضا را به او داد، تفاوت‌های بین ما خیلی زیاد است با وجود این به یک نقطه مشترک رسیدیم من نه با خودش با کارش هم ازدواج کردم» بابک می‌گوید: «من با این تضاد همیشه دست به گریبانم این که برای پیشرفت باید خودت را پرزنت کنی و در جمع باشی خیلی وقت‌ها مقاله می‌دهم وقتی موقع سخنرانی می‌شود پشیمان می‌شوم»

حنانه حاج سید جوادی در شاتل مهندس نرم‌افزار است که بعد به کیش ویر می‌رود و مدیریت پایگاه داده‌های اوراکل آن‌جا را در دست می‌گیرد. چهار سال بعد وقتی شرکت ایمن پردیس را راه‌اندازی می‌کنند به خواست بابک مدیرعامل شرکت می‌شود تا همچنان فقط کارهای فنی در دست بابک بماند. می‌گوید: «نظرش این بود که من آدم اجرایی هستم» می‌خندد: «شاید هم می‌خواست همه چیز را گردن من بیندازد». بابک می‌گوید: «من حوصله کارهای اجرایی را نداشتم و خودم را در این اتاق حبس کردم تا جایی که امکان داشته باشد سعی می‌کنم پای کامپیوترم باشم و خیلی با آدم‌ها سر و کله نزنم»

سال ۸۶ بابک فرخی تصمیم به نوشتن کتاب Network Administration with FreeBSD 7 می‌گیرد و

این کار یک سال طول می‌کشد. همسرش می‌گوید: «آن کار خیلی زمان بر و سخت بود یک سال از عمرش را گرفت ساعت هشت از شرکت می‌آمد و بعد از کمی استراحت تا چهار پنج صبح روی این کتاب کار می‌کرد و بعد دوباره هفت صبح می‌رفت. هدف هر دوی ما این بود که این کار انجام شود».

وقتی می‌پرسم چرا کتاب را به زبان انگلیسی نوشت می‌گوید: «این برای من یک چالش بود دنبال پول و معروفیت نبودم فقط دنبال این بودم که این کار دشوار را انجام دهم می‌دانستم خیلی کار سختی است». کتاب هنوز هم فروخته می‌شود و هر ماه انتشارات مبلغی را برای فرخی می‌فرستد. اصرار انتشارات برای نسخه دوم کتاب متناسب با فری‌بی‌اس‌دی ۱۰ مؤثر واقع نمی‌شود و بابک دیگر سراغ نوشتن کتاب نمی‌رود.

سلطان BSD

تجربه زیادش در کار با BSD و تولید بسته‌های متعدد برای این سیستم‌عامل باعث می‌شود یکی از توسعه‌دهندگان اصلی به او پیشنهاد بدهد که رسما توسعه‌دهنده بی‌اس‌دی شود و به CVS این پروژه دسترسی داشته باشد. او طی این مدت نزدیک به ۸۰ پکیج را نگهداری می‌کرد.

بعد از شاتل دوباره به داتک بر می‌گردد و چهار سال هم در آن‌جا می‌ماند. درباره این تغییر شغل‌های متعدد می‌گوید: «هر کجا کار روتین می‌شد دوست نداشتم می‌خواستم همیشه شیبی داشته باشد و شاتل خیلی سریع شیبش تمام شد» تصمیم می‌گیرد مشاور باشد و دورادور نظارت داشته باشد اما به اصرار شرکت دوباره معاون فنی می‌شود و چهار سال بعد متوجه می‌شود درگیر همان چیزی شده که نمی‌خواسته است. سال ۸۸ از داتک هم خارج می‌شود و برای دو سال با شرکت بیستاک کار می‌کند. می‌گوید: «بیستاک کار VoIP می‌کرد و این برای من یک فضای جدید بود» آن‌جا مدیر فنی می‌شود اما با این تفاوت که این بار برخلاف شاتل و داتک از ابتدای کار با شرکت نیست و ساختار فنی شرکت پیش از این درست شده است. می‌گوید: «وقتی من رسیدم قسمت فنی کار تمام شده و فقط نگهداری مانده بود» روی سرورهای آن‌جا فدورا نصب است اما به درخواست بابک فرخی و به خاطر رفتن مسئول سیستم‌ها، تمام آن‌ها پاک شده و فری بی‌اس‌دی می‌شود. دیدگاه مدیریتی و طبعا محافظه کارانه‌اش در نگهداری سیستم به خوبی نمایان است. می‌گوید: «من با فدورا هیچ وقت نتوانستم ارتباط برقرار کنم چون خیلی شکننده بود من از دنیای BSD و سولاریس به پایداری عادت داشتم و فدورا خیلی روی لبه فناوری بود تا چیزی را آپگرید می‌کردی سیستم خراب می‌شد من اگر بخواهم از لینوکس استفاده کنم حداقل دبیان استفاده می‌کنم»

آنجا هم خیلی زود شیبش تمام شد می‌گوید: «دیدم هرجا می‌روم شیبش تمام می‌شود گفتم چه کار کنم که تمام نشود» تصمیم می‌گیرد شرکت خودش را راه بیندازد. بنابراین شرکت ایمین پردیس را که دو سال قبل کارهای ثبتش را انجام داده بودند فعال می‌کنند. می‌گوید: «من اصلا به مدل استارت‌آپی اعتقاد ندارم این که یک نفر با ایده بیاید پیش سرمایه گذار بعد او هم پول بدهد یک دفتر بخرد با میز پینگ پونگ و فوتبال دستی این‌جا جواب نمی‌دهد. ما مشتری داشتیم کار هم داشتیم برای این که جنبه قانونی داشته باشیم شخصیت حقوقی‌مان را ثبت کردیم یک جایی باید به جمعیتمان اضافه می‌کردیم که مجبور شدیم دفتر بخریم همیشه اعتقاد داشتم هسته شرکت باید کوچک بماند اصلا هم دنبال رشد قارچی و فضای استارت‌آپی و سر و صدا نیستیم»

کارش را با مشاوره دادن شروع می‌کند بعد کلاس‌های متعددی برگزار می‌کند. می‌گوید: «اصلا فکر نمی‌کردم این‌قدر استقبال بشود من پنج‌شنبه جمعه‌ها هم می‌آمدم درس می‌دادم یهو از نیروی انتظامی زنگ می‌زدند که می‌خواهیم ۲۰ نفر را بفرستیم BSD یاد بگیرد» شهرستانی‌ها در روزهای تعطیل درخواست بوت‌کمپ می‌کنند و هر شرکتی که می‌خواهد به کارمندانش سولاریس یا BSD یاد دهد اول سراغ بابک فرخی می‌رود. تجربه چندین ساله کار در شرکت‌های مختلف برای او دستاورد بزرگی شده است. کار آن‌قدر سنگین و انرژی بر است که بعد از اجرای موفق چندین دوره آموزشی آن را متوقف می‌کند. می‌گوید: «آموزش را خیلی دوست دارم هر از چندگاهی سعی می‌کنم بچه‌های شرکت را جمع کنم به آن‌ها چیزهای جدیدی را که آشنا شده‌ام یاد بدهم».

علت استفاده نکردنش از لینوکس دلایل متفاوتی دارد. گاهی کاملا فنی است و گاهی اخلاقی. از نظر او لینوکس به نسبت BSD بیش از حد ناپایدار و آشفته است و برای سرور مناسب نیست. دلیل دیگرش را این طور می‌گوید: «من یکی از ناراحتی‌هایم این است که این آقای توروالدز خیلی بد دهن و بی‌تربیت است اصلا خوشم نمی‌آید. در BSD یک سری آدم جنتلمن باشخصیت اصلا نه حاشیه دارند نه برای مطرح کردن خودشان به کسی فحش می‌دهند. مثلا دکتر مارشال کک مک‌کیوستیک و آقای جرج نویل و آقای دکتر رابرت واتسون کتابی به نام طراحی و پیاده‌سازی سیستم‌عامل فری بی‌اس‌دی دارند. کتاب‌هایشان را می‌خوانی لذت می‌بری من چون با این سه نفر در ارتباطم بسیار جنتلمن و اهل کمک هستند تو سؤال احمقانه بپرس با حوصله جواب می‌دهد» ادامه می‌دهد: «سیستم بی‌اس‌دی را بیشتر می‌پسندم که قائل به شخص نیست و سیستم دموکراتیک است در لینوکس، توروالدز با آلن کاکس دعوایش می‌شود و بهبودی را که کاکس در هسته انجام داده را چون با هماهنگی او نبوده بر می‌گرداند و دوباره همان اشکال بد را می‌گذارد».

از نظر او لایسنس بی‌اس‌دی بهتر از جی‌پی‌ال است. در این باره می‌گوید: «چون وقتی می‌گویی آزاد یعنی آزاد، جی‌پی‌ال آزادی مشروط دارد که از نظر من جالب نیست. بی‌اس‌دی می‌گوید اگر اسرار تجاری داری سورس کد را منتشر نکن ما زورت نمی‌کنیم برای همین سونی و خیلی‌های دیگر از آن استفاده می‌کنند با این حال اکثر شرکت‌هایی که از کدهای تحت لایسنس بی‌اس‌دی استفاده می‌کنند روی همان پروژه‌ها مشارکت می‌کنند و مشکلی با کانتریبیوت کردن ندارن و حتی ریپوزیتوری فورک شده ندارند اما این لایسنس حالشان را خوب می‌کند»

مشخص است که تمام انتخاب‌هایش تخصصی و به دور از تعصب است. می‌گوید: «بیشتر از ده سال است که زندگی‌ام در مک است و ویندوز را به خاطر آفیسش دارم فن ویندوز نیستم اما معتقد هم نیستم ویندوز آشغال است لزوما از نرم‌افزارهای متن‌باز و آزاد استفاده نمی‌کنم یکسری نرم‌افزار تجاری دارم که روی ویندوز و مک است و می‌توانم خودم را وفق دهم اما نمی‌آیم خودم را عذاب بدهم می‌روم ۲۰ دلار پول می‌دهم تا پست‌باکس را جای تاندربرد استفاده کنم. هیچ وقت روی میزکارم BSD نصب نمی‌کنم چون مک بهتر است و قطعا هیچ وقت نسخه سرور مک یا ویندوز را روی سرورم نصب نمی‌کنم، قطعا اوراکل را روی BSD نصب نمی‌کنم NetBSD روی ۳۰ پلتفرم مختلف اجرا می‌شود این‌ها هرکدام قطعات پازل هستند که باید در جای خودش بگذاری».

از آپادانا تا ایمن پردیس

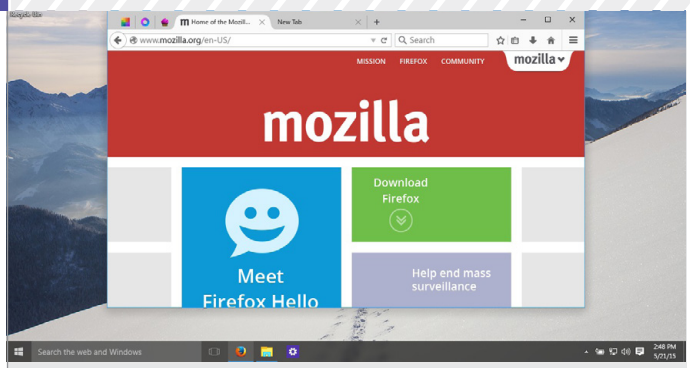
بابک فرخی حالا مدیر فنی شرکت «ایمن پردیس» است. شرکتی که محصول «فرعی‌اش» یعنی «بیسفون» معروف‌ترین اپلیکیشن ارتباطی ایرانی است. خیلی‌ها این نرم‌افزار، رسانه اصلی‌شان شده و برای خود صفحه رسمی ساخته‌اند. تعدادی از

کانال‌ها روزی ۴ هزار دنبال کننده جدید دارند. بیسفون ۴ سال پیش بر اساس یک ایده بلند پروازانه شکل گرفت، آن را با شرکت بیستاک در میان گذاشتند و شروع کننده یک جریان جدید شدند. می‌گوید: «مدیر بیستاک هم انسان جسوری بود و از این که به ما بخندند نمی‌ترسید با چراغ کاملا خاموش کاری را شروع کردیم که در آن تخصصی نداشتیم چون نرم‌افزار بک اِند و سروری خوب تولید می‌کنیم اما در حوزه موبایل اصلا. رفتیم سراغ کسانی که بلد بودند اما دیدیم آن‌هایی که ادعا می‌کردند خیلی در این حوزه سوار کار نیستند برای همین خودمان آستین‌ها رو بالا زدیم.» او حتی برای این کار خودش برنامه نویسی موبایل یاد می‌گیرد و شروع به کدنویسی می‌کند. می‌گوید: «خود من از اسمارت فون فراری بودم اما با Objective-C روی اپل کد نوشتم الان هم تیم ۱۸ نفره تقریبا تمامشان از این جا شروع کردند». درباره جو شرکت ایمن پردیس می‌گوید: «اتمسفر ما خیلی اتمسفر خاصی است همه با آغوش باز می‌خواهند به هم چیز یاد بدهند».

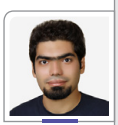
حالا بدون این که از بیسفون درآمدی داشته باشند به خوبی در حال توسعه آن هستند. می‌گوید: «بیسفون به عنوان سرگرمی شروع شد و هنوز هم همین‌طور است و به شدت هم برایش هزینه می‌کنیم ولی پولمان را از جای دیگری در می‌آوریم البته بیسفون هم قرار است پول درآورد.» کار اصلی شرکت فروش یک سیستم CGN است که در سطح Carrier عملیات Nat انجام می‌دهد. اپراتورهای موبایل و شرکت‌های مهمی از این سرویس استفاده می‌کنند. کار این محصول این است که برای مهاجرت به IPv6 زمان می‌خرد. می‌گوید: « مهاجرت به نسخه ۶ زمان‌بر است برای این زمان جابجایی اپراتورها دچار کمبود آی‌پی هستند و باید نَت کنند منتهی نت برای کاربر نهایی طراحی شده نه در مقیاس بزرگ. ما محصولی درست کردیم که در ابعاد بزرگ می‌تواند نت انجام دهد بدون این که سرویس کاربر را تحت تأثیر قرار دهد و این یعنی میلیون‌ها نشست و چند ده میلیون بسته در ثانیه بدون این که کیفیت پایین بیاید در دنیا این کار انگشت شمار است» در ۱۸ سال گذشته کار اصلی بابک فرخی شبکه و برنامه نویسی بوده در حوزه‌هایی هم مثل روتینگ شبکه آشنایی پیدا کرده است.

یکی از محصولات شرکت Net OS نام دارد که یک سیستم‌عامل مبتنی بر FreeBSD است و برای کارهای شبکه و روتینگ بهینه شده است. یک فلش کوچک را میان دو انگشتانش می‌گیرد و به ما نشان می‌دهد. می‌گوید: «اگر از اینترنت رایتل استفاده می‌کنید از این رد می‌شود» این سیستم‌عامل برای پردازنده‌های زئون اینتل بهینه شده تا از دستورالعمل‌های بیشتری پشتیبانی کند و مثلا محاسبات وکتور به جای نرم‌افزار درون پردازنده پردازش شود. به ترمینالی که روی سیستمش باز است اشاره می‌کند و می‌گوید: « ۳ میلیون بسته در ثانیه را روی هر دستگاه فورووارد می‌کند.»

مصاحبه تمام شد. عکاس جدیدی که آمده بود هم عکس‌هایش را گرفت و آماده رفتن شدیم. سیگاری دود کرد و تعارفی زد که مرا با موتورش تا جایی ببرد. عکاس خبرگزاری مشرق است و در مهر و فارس هم کار کرده. ترک موتورش که نشسته بودم گفت: «آدم گاهی می‌ماند ما با این همه ادعای حزب‌اللهی بودن هیچ کاری برای مملکت نکردیم و این‌ها بدون هیچ ادعایی چقدر برای کشور مفید بودند» از صداقتش خوشم آمد. گفتم: «نه فقط برای ایران برای کل جهان». گفت: «برای کل جهان».■



نامه سرگشاده مدیرعامل موزیلا به مدیرعامل مایکروسافت



مترجم: دانیال بهزادی

ساتیا: من دارم درباره یکی از جنبه‌های بسیار مخرب ویندوز ۱۰ به تو نامه‌نگاری می‌کنم. به‌طور صریح‌تر این که به نظر می‌رسد تجربه به‌روز رسانی به گونه‌ای طراحی شده که انتخابی که کاربرانتان درباره تجربه اینترنتی‌ای که می‌خواهند کرده‌اند را دور انداخته و با تجربه اینترنتی‌ای که مایکروسافت می‌خواهد داشته باشند جایگزین کند. هنگامی که نخستین بار تجربه به‌روز رسانی ویندوز ۱۰ را دیدیم که کاربران را با پایمال کردن موثر ترجیحات کاربری موجودشان برای مرورگر وب و دیگر کارهایشان، از انتخاب‌های خود بازمی‌دارد، با تیم شما تماس گرفتیم تا در مورد این مشکل بحث کنیم. متأسفانه این امر منجر به هیچ پیش‌رفت معنی‌داری نشد.

ممنونیم که هنوز به صورت فنی این امکان وجود دارد که تنظیمات و پیش‌گزیده‌های پیشین افراد نگاه داشته شود، ولی طراحی تجربه کلی به‌روز رسانی و تنظیمات و رابط‌های برنامه‌نویسی پیش‌گزیده به‌گونه‌ای تغییر یافته که این کار را مبهم‌تر و سخت‌تر کند. حالا افراد برای برگرداندن انتخاب‌هایی که پیش‌تر در نگارش‌های قدیمی‌تر ویندوز ایجاد کرده بودند نیاز به بیشتر از دو بار کلیک موشی، پیمایش در میان محتویات و مقداری مهارت فنی دارند. این کار گیج‌کننده و دشوار است و به آسانی قابل از بین رفتن.

فلسفه وجودی موزیلا، آوردن انتخاب، کنترل و فرصت برای همه است. فایرفاکس و دیگر محصولاتمان را برای این منظور ساخته‌ایم. موزیلا را به عنوان یک سازمان غیرانتفاعی برای این منظور ساخته‌ایم. و کار می‌کنیم تا تجربه اینترنتی، ورای محصولاتمان تا جایی که می‌توانیم این ارزش‌ها را ارائه دهند.

گاهی پیش‌رفت‌های بزرگی می‌بینیم، جایی که محصولات مصرفی به افراد و انتخاب‌هایشان احترام می‌گذارند. با این حال، با به بازار آمدن ویندوز ۱۰ عمیقا ناامید شدیم که دیدیم مایکروسافت چنین گام چشم‌گیری رو به عقب برداشته است.

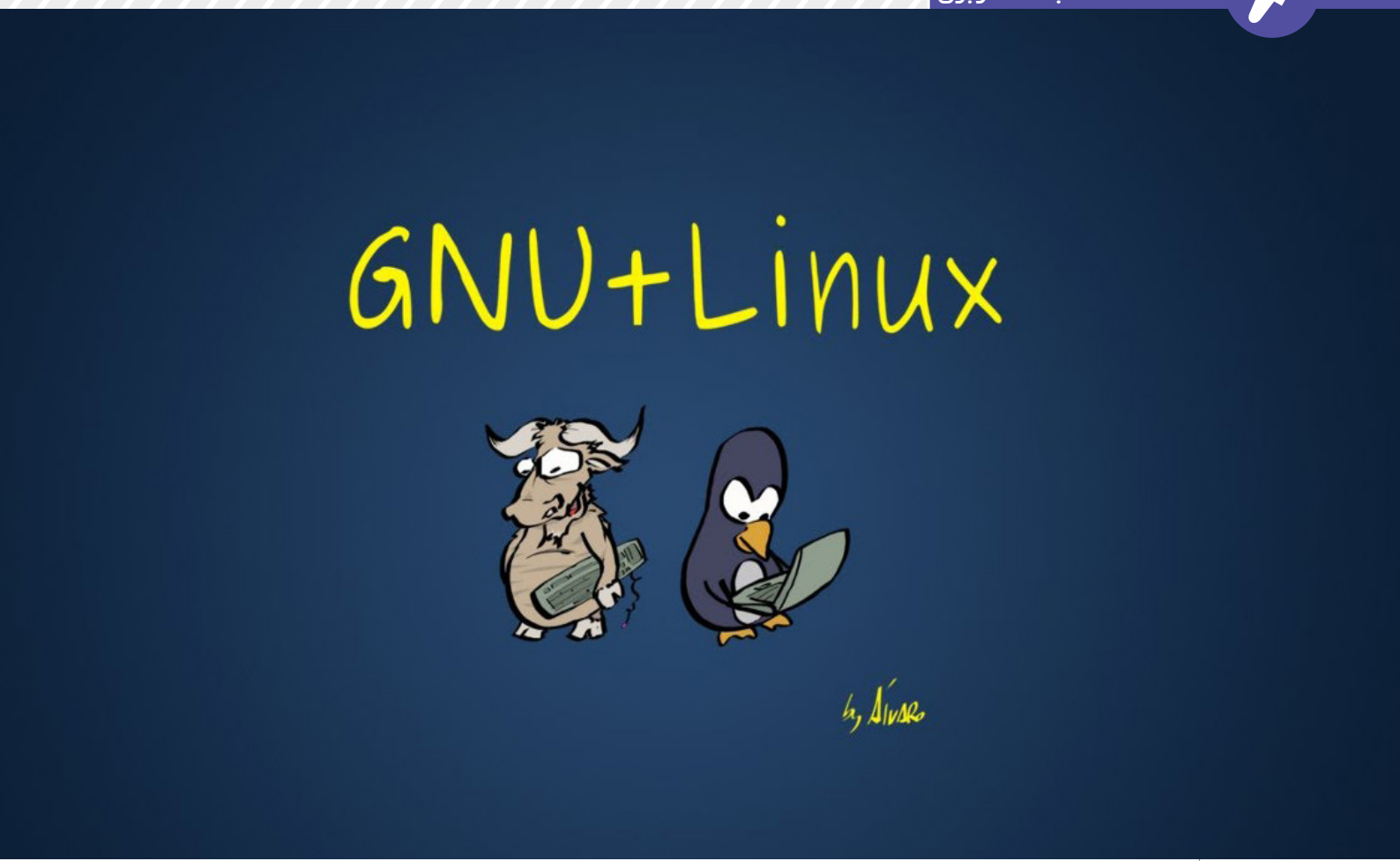
این تغییرات ما را به دلیل این که سازمانی هستیم که فایرفاکس را می‌سازد ناراحت نمی‌کند. این‌ها به این دلیل ناراحت‌کننده هستند که میلیون‌ها کاربر، ویندوز را دوست دارند و همان‌هایی هستند که انتخاب‌هایشان نادیده گرفته شده است، و به دلیل پیچیدگی افزوده شده‌ای که در راه هرکسی که هرگاه بخواهد انتخابی متفاوت از آن‌چه مایکروسافت می‌پسندد داشته باشد، گذاشته شده است.

قویا به شما توصیه می‌کنیم که تاکیک تجاری خود را مورد بازنگری قرار داده و با ساده‌تر، واضح‌تر و شهودی کردن حفظ انتخاب‌هایی که مردم تا کنون کرده‌اند در تجربه به‌روز رسانی، دوباره به حق آن‌ها برای انتخاب و کنترل آسان‌تر تجربه برخطشان احترام بگذارید. تنظیم انتخاب‌ها و ترجیحات جدید باید برای مردم ساده‌تر باشد، نه فقط برای محصولات دیگر مایروسافت، بلکه از طریق رابط‌های برنامه‌نویسی تنظیمات پیش‌گزیده و رابط‌های کاربری.

لطفا در ویندوز ۱۰، انتخاب و کنترلی که کاربرانتان مستحق آن هستند را بهشان بدهید.

خالصانه،

کریس بیرد|مدیرعامل موزیلا



بهانه‌های استفاده نکردن از گنو/لینوکس

نویسنده:
بروس بایفیلد

در زمانی که گنو/لینوکس تازه معروف شده بود، مقالات مختلفی کمبودهای گنو/لینوکس را محکوم می‌کردند. به سختی ممکن بود ماهی بگذرد، بی آن که کسی توضیح بدهد گنو/لینوکس چه کم دارد، یا برای قابل استفاده بودن به چه ویژگی، برنامه یا سرویسی نیاز دارد. و مثل اغلب موارد، شکایت‌ها گمراه کننده بودند.

مسلما نرم‌افزارهای آزادی که روی گنو/لینوکس اجرا می‌شوند، مثل تمام نرم‌افزارهای دیگر دارای برخی کاستی‌ها و نواقص هستند. مثلا کماکان شما نمی‌توانید فرم‌های پی‌دی‌اف را پر کنید، یا در برخی کشورها مالیات خود را با گنو/لینوکس محاسبه کنید. در موارد دیگر، مثل تشخیص نویسه نوری یا تشخیص گفتار نرم‌افزارهای آزاد وجود دارند، ولی مقابل رقبای انحصاری ابتدایی به نظر می‌رسند. البته هر ساله کمبودها و نقایص کم‌تر و کم‌تر می‌شوند و شکایت‌ها اغلب به خاطر ناآگاهی است تا هر چیز دیگر.

اما چه نوع از ناآگاهی؟ این‌جا جایی است که بحث جذاب می‌شود.

آرزوی ادامه استفاده از نرم‌افزار آشنا

تعویض سیستم‌عامل یک مهاجرت عظیم است و دوره‌ی گذار آن قابل اجتناب نیست. با این حال بسیاری از کاربران بالقوه در صورتی اقدام به مهاجرت می‌کنند که بتوانند به استفاده از نرم‌افزارهای همیشگی خود ادامه بدهند.

احتمالا دلیل این نگرش، این است که محصولات اصلی شرکت‌هایی چون ادوبی و مایکروسافت، روی سیستم‌عامل‌های ویندوز و اواس ۱۰ یکسان‌اند. بیشتر کاربران پس از مهاجرت فقط تفاوت‌های دو سیستم‌عامل را می‌آموزند. پس انتظار دارند اوضاع در گنو/لینوکس هم، با وجود تاریخچه و اهداف متفاوت مشابه باشد.

دقت نکردن به ویژگی‌ها

افراد انتظار دارند گنو/لینوکس در برابر ویندوز و اواس ۱۰ بدوی و ابتدایی باشد، در نتیجه زمانی که یک ویژگی در فهرستی که انتظارش را دارند قرار نگرفته باشد یا نام متفاوتی داشته باشد، به جای جست‌وجو، استدلال می‌کنند که اصلا

وجود ندارد.
این نگرش به حدی قوی است که بخش صفحات گسترده لیبره‌افیس، احساس کرد که باید DataPilots را به Pivot Tables تغییر نام بدهد تا با اکسل مایکروسافت هم‌گام شود. البته اغلب اوقات، کاربران چیزی را که در انتظارش هستند می‌یابند.

عدم آگاهی از جایگزین

نرم‌افزارهای آزاد به خاطر این که توسط انگیزه‌های اقتصادی محدود نشده‌اند، معمولا دارای چندین جایگزین در هر گروه هستند. در اغلب موارد کاربران جدید از تمام این جایگزین‌ها آشنا نیستند.
برای مثال، یک کاربر جدید ممکن است گیمپ را به دلیل عدم پشتیبانی مستقیم از مدل رنگ سی‌ام‌وای‌کی یا تجزیه رنگ، به عنوان جایگزینی برای فوتوشاپ نپذیرد. ولی شروع به جست‌وجو نمی‌کند تا دریابد کریتا هردوی این کارها را انجام می‌دهد. در عوض کاستی‌های گیمپ را به عنوان اشکال نرم افزارهای گرافیکی گنو/لینوکس توصیف می‌کند.

موجود نبودن یک سرویس

گاهی اوقات کاربران تاکید می‌کنند که به پشتیبانی از یکی از خدمات برخط بر روی گنو/لینوکس نیاز دارند. خدمتی که اخیراً بیشتر به آن اشاره می‌شود گوگل درایو است.
در واقع راه‌حل‌هایی برای داشتن گوگل درایو روی گنو/لینوکس وجود دارد، اگرچه برخی از آن‌ها خیلی موقتی به نظر می‌رسند. ولی حتا اگر این راه‌کارها هم وجود نداشته باشند، چرا اصرار بر گوگل درایو دارید؟ این تنها خدمات ابری موجود نیست.

فرضیات مبتنی بر اطلاعات نادرست

رایج‌ترین مثال برای این بهانه، ادعایی است که می‌گوید گنو/لینوکس فاقد راهی آسان برای نصب نرم‌افزارهای جدید است. در عوض مدعی هستند که شما باید همه نرم‌افزارهایتان را از کد مبدأ کامپایل کنید. این بهانه وقتی جذاب می‌شود که همان افرادی که بهانه را ساخته‌اند مشکلی با نصب نرم افزار در گوشی تلفنشان که همان تجربه‌ی موجود در گنو/لینوکس را برای نصب نرم‌افزار ارائه می‌دهد، ندارند.

فرضیات مبتنی بر اطلاعات قدیمی

عمده دلایلی که از استفاده گنو/لینوکس توسط کاربران جلوگیری می‌کنند، مبتنی بر اطلاعاتی هستند که زمانی درست بوده‌اند، ولی سال‌ها است که غلط هستند.
نمونه‌هایی از این دلایل عبارتند از: شما باید هر چیزی را از طریق خط فرمان اجرا کنید، نصب آن دشوار است و پشتیبانی‌ای در کار نیست. تمامی این دلایل بیش از یک دهه است که دیگر درست نیستند.

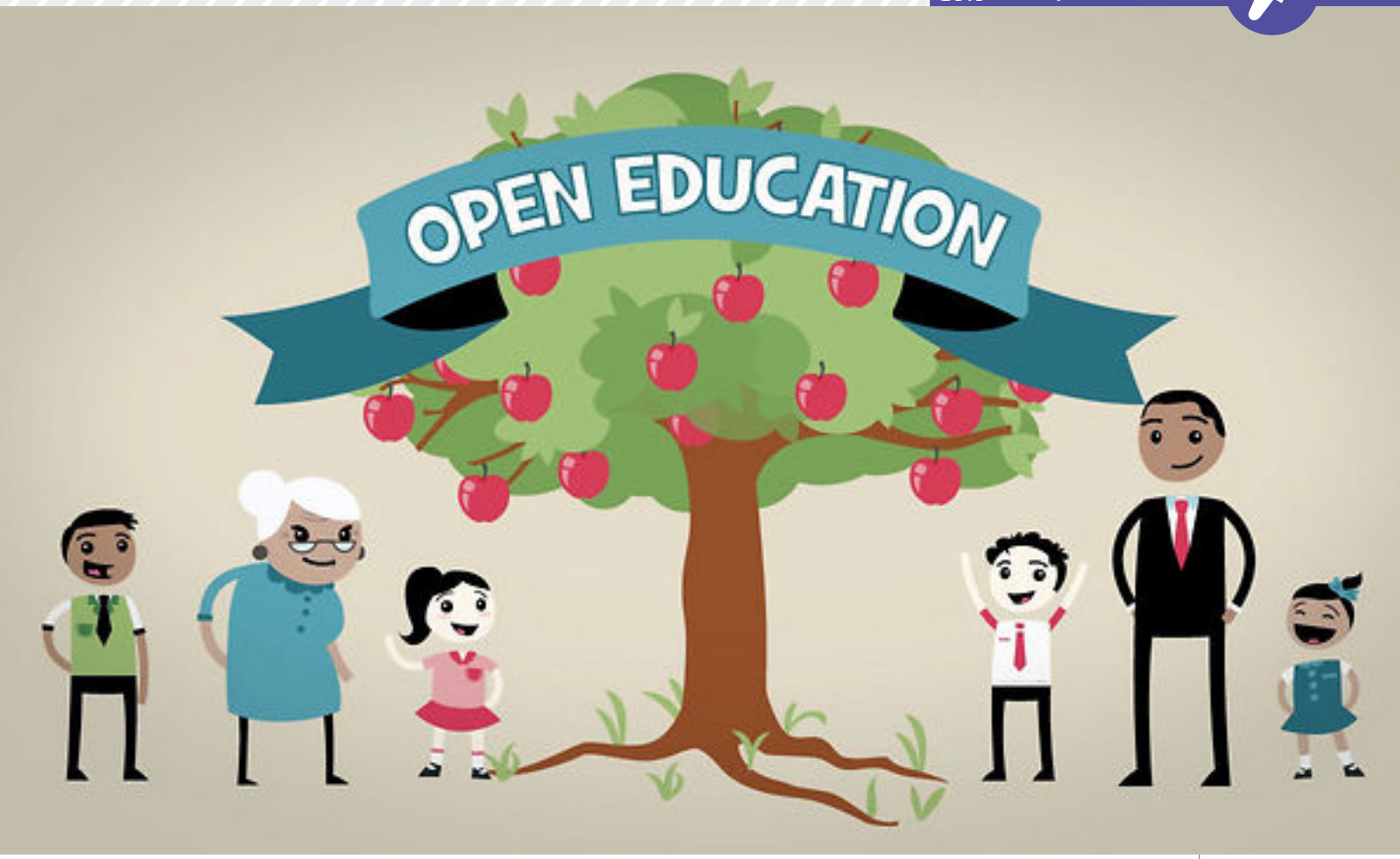
کمبود ابزارهای انحصاری

برای سال‌ها تازه واردان می‌گفتند چیزی که گنو/لینوکس واقعا نیاز دارد، ابزارهای انحصاری استاندارد است، که بیشتر اوقات منظورشان فوتوشاپ بود. باید توجه داشته باشیم که برای بسیار از افراد، نکته اصلی در نرم‌افزارهای آزاد استفاده نکردن از ابزارهای انحصاری است. البته مقصر، تولیدکنندگان نرم‌افزاری هستند. این تولیدکنندگان همچنان به دنبال یافتن فهمیدن روش پول درآوردن از طریق فروش نرم‌افزار هستند و پس از تلاش‌هایی مانند فریم‌میکر ادوبی در سال‌های ۲۰۰۰ و ۲۰۰۱ به این نتیجه رسیده‌اند که نمی‌توانند.
به هر حال ده سال پیش، فروشگاه‌های نرم‌افزاری آزاد به خاطر راحتی کاربران، ابزارهای انحصاری را هم می‌پذیرفتند. اما به خاطر نداشتن پشتیبانی، شروع به ایجاد نرم‌افزارهای جایگزین کردند. برای مثال امروزه دیگر کسی وجود ندارد که با وجود لیبره‌آفیس، خود را برای استفاده از فریم‌میکر به دردسر بیندازد. در بسیاری از موارد اگر هنوز اندک شانسی برای نرم‌افزارهای انحصاری روی گنو/لینوکس وجود داشته باشد نیز، احتمالا در چند سال آینده توسط نرم‌افزارهای آزاد جایگزین گرفته خواهد شد.

مغالطه‌های بی‌پایان

روشی که افراد در استفاده از این بهانه‌ها دارند مرا به یاد استدلال‌های خلقت‌گرایان درباره‌ی پیدا نشدن فسیل‌های حلقه‌ی مفقوده می‌اندازد. تکامل‌گرایان اغلب به فرم‌های انتقالی اشاره می‌کنند، در ادامه خلقت‌گرایان از آن‌ها درخواست فرم‌های انتقالی بین آن فرم‌های انتقالی را می‌کنند تا جایی که طرف مقابل از این بحث فرسایشی خسته شده و به آن پایان دهد. آنگاه خلقت‌گرایان اعلام پیروزی می‌کنند.
به طریق مشابه، طرفداران نرم‌افزار آزاد می‌توانند شواهدی را علیه این بهانه‌ها ارائه کنند. در واکنش به این شواهد، بهانه‌سازان به جای عقب‌نشینی از موضعشان، همان ایراد را به گروه دیگری از نرم‌افزارها اعمال می‌کنند. تا زمانی که دیگر طرفداران نرم‌افزار آزاد بحث با چنین کسی را بی‌فایده ببینند. آن موقع است که بهانه‌سازان ادعای پیروزی می‌کنند و متقاعد می‌شوند که اجتناب آن‌ها از گنو/لینوکس کاملاً توجیه شده است.
البته این بهانه‌ها در مورد انتظارات معقول نیستند، حتا در مورد گنو/لینوکس و نرم‌افزار آزاد هم نیستند. جز در موارد معدودی که کاستی‌هایی وجود دارد، این بهانه‌ها فقط توجیه‌هایی برای درست نشان دادن خود هستند. شک دارم چیزی که بهانه‌سازان بیان می‌کنند، واقعا آرزویی برای جلوگیری از آزار دیدنشان توسط آزادی نرم‌افزار و راحت گذاشتنشان برای انجام رایانششان به روش‌هایی که همیشه انجامش می‌داده‌اند باشد.■

عمده دلایلی که از استفاده گنو/لینوکس توسط کاربران جلوگیری می‌کنند، مبتنی بر اطلاعاتی هستند که زمانی درست بوده‌اند، ولی سال‌ها است که غلط هستند



آشنایی با استادی که باعث صرفه‌جویی زیادی برای دانشجویان شد

صرفه‌جویی میلیون دلاری و دانش آزاد

نویسنده: نیکو رائگارد

در ابتدا با دیوید لیمپ‌من از طریق مقاله‌ای که در TeamOpen منتشر کرده بود، آشنا شدم؛ و بعدازآن بود که احساس کردم که باید با او در مورد کارش در آموزش‌های آزاد و متن‌باز صحبت کنم. دیوید یک استاد در کالج پیرس است؛ که باعث شده است دانشجویانش میلیون‌ها دلار به دلیل استفاده از کتاب‌های اشتراکی و دانش آزاد، صرفه‌جویی مالی داشته باشند. او همچنین برنامه‌ای متن‌باز و آزاد به نام IMathAS را توسعه داده است، که برای آموزش ریاضی و پلتفرم آموزشی، تهیه‌شده است.

کمی در باره خودتان بگویید چگونه به دنیای متن باز و آموزشی وارد شدید؟

یک معلم حرفه‌ای ریاضی هستم، با این حال سرگرمی دیگری را نیز همیشه دنبال می‌کنم؛ این سرگرمی‌ها برنامه‌نویسی و کامپیوتر هستند. اما برنامه‌نویسی و رایانه برایم همیشه یک فعالیت ذهنی فعال بوده‌اند. در طول سال‌ها، از چندین برنامه متن‌باز استفاده کرده‌ام، و یا به پروژه‌های کوچکی پیوسته‌ام، مانند ASCIIMathML. در حدود ۹ سال پیش، هنگامی که انجام تکالیف به‌صورت اینترنتی به جریانی اصلی مبدل شد؛ با خود گفتم که این موضوع واقعا ایده‌ی خوبی است، اما نمی‌توانم چنین هزینه‌ای را به دانشجویان تحمیل کنم. طبیعتا دنبال این رفتم تا گزینه‌های متن‌باز و ارزانی را به‌جای موارد فعلی استفاده نمایم. در ابتدا با پروژه WebWork آشنا شدم. اما بعد متوجه شدم تنظیم آن بسیار پیچیده است. این‌زمانی بود که من IMathAS را به وجود آوردم؛ تا بسیار راحت‌تر قابل تنظیم باشد. این برنامه پروژه‌ای بود که به‌راحتی با LAMP کار می‌کرد به این خاطر که من علاقه داشتم از پروژه‌های متن‌باز استفاده کنم. پس‌ازآن و با گذشت زمان، وقتی که این سیستم به خوبی پیاده‌سازی شد؛ به استفاده از آن اقدام کردیم،

سپس بعد از چند سال نظرم به این موضوع جلب شد و با خود گفتم، که چرا هر فرد باید ۱۵۰ دلار پول کتابی را بپردازد که بعدا ممکن است به آن نیازی نداشته باشد؟ این‌جا بود که بر روی مفاهیم کتب متنی باز و آموزش آزاد (باز) تمرکز کردم. پروژه کتابخانه دروس آزاد در واشنگتن به من و همکارانم زمانی را برای کار بر روی این ایده‌ها قرار دادند تا به‌کار بر روی آنان بپردازیم. همچنین با همکاری با دیگر دانشگاه‌ها و مدارس، می‌توانستیم به مواد دست اول و عالی، موردنیاز دسترسی داشته باشیم

iMathAs، را چگونه ارزیابی می‌کنید؟

بعدازاین که اولین نسخه را ایجاد کردم. آن را به فرد دیگری از جامعه دانشجویان کالج در ایالتمان نشان دادم. او گفت که "فکر می‌کنی که پروژه قرار است چه گونه پیش رود؟ " و به او گفتم که می‌خواهم این پروژه در کل ایالت قابل‌دسترسی باشد. سپس او رفت و با بودجه مالی برای انجام پروژه و ایجاد میزبانی اینترنتی برای قرار دادن آن در بستر وب، بازگشت. به همین خاطر اکثر دانشکده‌ها در سراسر ایالت شروع به استفاده از این برنامه کردند. در ابتدا فقط دانشکده‌های بزرگ و آموزشکده‌ها به استفاده از آن پرداختند. مانند کسانی که نمی‌خواستند هزینه‌های زیادی را مانند چیزی که فعلا پرداخت می‌کردند، پرداخت کنند، اما بعدا این برنامه توسط عده‌ی بیشتری، حتی افراد حقیقی نیز مورداستفاده قرار گرفت.

بعدازآنکه تعداد زیادی دوره توسط کتب متنی انجام شد، بسیار مشتاق بودم این کتاب‌ها و ابزار را در خارج از واشنگتن نیز منتشر نمایم. بنابراین یک دامنه با آدرس myopenmath.com راه‌اندازی کردم تا این گزینه در خارج از ایالت نیز قابل‌دسترسی باشد. تعداد کمی کالج وجود داشت که تصمیم گرفتند؛ نمونه‌های خود را میزبانی کنند، اما اکثر مردم استفاده از خدمات میزبانی‌شده را بسیار راحت‌تر یافتند. خوشبختانه ما توانستیم با یک شرکت تجاری همکاری کنیم که برای ما پشتیبانی فنی و نرم‌افزاری را انجام می‌دهد و همچنین هزینه‌ها را هم متقبل شد. در پاسخ به سوال شما، به‌طورکلی آن را خوب ارزیابی می‌کنم، تعدادی دانشکده هستند که در حال حاضر؛ از این برنامه بهره می‌برند. با این حال ما به بیش از ۳۰۰۰۰ دانشجو در هر ترم، خدمات می‌دهیم.

ابزار آموزشی متن باز مورد علاقه شما چیست؟

من واقعا نمی‌دانم چه چیزهایی واقعا به‌عنوان ابزار آموزشی باز (آزاد) هستند، اما واقعا به وب‌سایت UWN علاقه دارم. و همچنین کاری که NC Campus و Lumen برای توسعه Press-book انجام داند.

ابزار متن باز مورد علاقه شما چیست (به جز ابزار خودتان)؟

از خیلی از آنان استفاده می‌کنم. jEdit ابزار اصلیم برای کد نویسی است. همچنین از Gimp و Inskcape برای ویرایش تصاویر و طراحی استفاده می‌کنم. MathJax و jsxGraph هم کتابخانه‌های بسیار خوبی هستند، که در پروژه‌ی خودم از آنان استفاده می‌کنم.

درباره پیشرفت کارتان چه نظری دارید؟ و آیا به نظر شما، این پروژه می‌تواند به تعداد افراد بیشتری منفعت برساند؟

این خیلی باحال است، علاقه‌مند هستم تا افراد بیشتری آن را استفاده کنند اما ما هیچ‌وقت فکرش را نمی‌کردیم که تعداد افرادی که از این برنامه استفاده می‌کنند، تا این حد پیشرفت کنند. این واقعا جالب است؛ وقتی‌که می‌بینیم برخی افراد به ما پیغام‌هایی را ارسال می‌کنند، که با ما نشان دهند، استفاده از این برنامه را آغاز کرده‌اند. درهرصورت با استفاده از گسترش‌دهنده‌های OpenStax، این امید وجود دارد که برنامه ما به‌صورت گسترده‌تری موردپذیرش قرار گیرد.

اینجا بود که بر روی مفاهیم کتب متنی باز و آموزش آزاد (باز) تمرکز کردم. پروژه کتابخانه دروس آزاد در واشنگتن به من و همکارانم زمانی را برای کار بر روی این ایده‌ها قرار داند تا به‌کار بر روی آنان بپردازیم.

آیا تا به حال این موضوع را در نظر گرفته‌اید که از سیستم کنترل نسخه‌ای مانند gitHUB برای پروژه خود استفاده کنید؟

ما تصمیم گرفتیم که کتاب‌های خود را در Word بنویسیم. به این خاطر که مخاطبان و اکثر جامعه‌ی دانشگاهی و آموزشکده‌ها بسیار بیشتر از TeX به Word علاقه داشتند. درحالی‌که ما می‌توانیم به‌راحتی از GitHUB استفاده کنیم، حتی برای قرار دادن فایل Word. ولی از آن استفاده نخواهیم کرد؛ شاید افراد نتوانند از GitHUB استفاده کنند؛ بنابراین ما آرشیوی را در وب‌سایتمان قرار داده‌ایم تا به‌راحتی به تمام نسخه‌ها دسترسی داشته باشند. درهرصورت اگر نسخه‌ای HTML از کتب متنی آماده شد آنان را در GitHUB قرار خواهیم داد.

آیا پیشنهاد خاصی برای مربیان آموزشی دارید تا وارد جنبش آموزش باز (آزاد) شوند؟

برای آداپتورها، امکانات بسیاری بیشتری نسبت به آن‌چه در سابق بود، وجود دارد. و به‌راحتی می‌توان گام‌های زیادی را به جلو برداشت بدون آن‌که وقت زیادی را در این زمینه هدر داده باشید. توصیه می‌کنم که به‌عنوان مهمان ابتدا وارد سایت MyOpen-Math.com شده و ببینند باکمی تلاش؛ چه چیزهایی به‌دست‌آمده است. همچنین برای دانشگاه‌ها در رشته‌های دیگر پیشنهاد می‌کنم تا از پروژه‌های کتب متنی باز، OpenStax و آموزش Lumen و.. دیدن نمایند تا ببینند چه چیزهایی در این سایت‌ها قرار دارد.■

آیا نرم‌افزار امنیتی آزاد امنیت پایینی دارد؟

این سوالی تکراری است که ما معمولا در بنه‌تک «Benetech» در ارتباط با مارتوس می‌شنویم. ابزار قدرتمند و رایگان ما برای رمزنگاری، برای امن کردن جمع‌آوری و مدیریت اطلاعات حساس است که توسط برنامه حقوق بشر بنه‌تک[1] ایجاد و آماده شده است. سوال مهمی است برای ما و همه کسانی که توسعه‌دهنده نرم‌افزارهای متن‌باز/آزاد هستند؛ آن هم در عصر پسا-اسنودن که کاربران زیادی نگران درز اطلاعات شخصی یا شنود اطلاعاتشان توسط سرویس‌های امنیتی هستند. با این حال ما بر این باوریم که نه تنها نرم‌افزارهای آزاد امنیت بالایی دارند بلکه برای آن ضروری هستند.

اجازه دهید تا قیاس زیر را تعریف کنیم:

رمزنگاری ترکیبی امنیتی است که باعث حفاظت از اطلاعات شما می‌شود و شما تنها کسی هستید که از آن ترکیب استفاده می‌کنید. یا این که این عمل را به یکی از همکارانی که به شما نزدیک است، می‌سپارید که هدف شما دور نگاه‌داشتن افراد غیرمجاز از دسترسی به داده‌هایتان است. آن‌ها ممکن است سارقینی باشند که قصد دزدیدن اطلاعات مرتبط با کسب و کار شما را دارند یا این که کارمندان همرده شما که قصد اطلاع از حقوقتان را دارند. یا موکلی که قصد دزدیدن اطلاعات مرتبط با کلاهبرداری را دارد؛ در تمام این موارد شما مجبورید اطلاعات خود را از آنان محفوظ داشته و از دسترسی افراد غیرمجاز جلوگیری کنید.

حال اجازه دهید بگویم مکانی امن برای اشیاء با ارزشم یافته‌ام. دو گزینه پیش رو دارم گزینه اول که از گاوصندوقی در تبلیغات مطلع می‌شوم که نیم اینچ عمق دیواره آن بوده و یک اینچ ضخامت درب آن است. همچنین برای نگه‌داری اطلاعات از ۶ قفل پیچیده برخوردار است. که توسط آژانس امنیتی مستقلی تایید شده است که تا دو ساعت در آتش دوام می‌آورد. گزینه دوم این است که از گاوصندوقی استفاده کنم که سازنده آن گفته است روش امن نگاه‌داشتن چیزهایی که درون است به صورت سری است که می‌تواند اطلاعات شما را با تخته سه‌لا و استیلی نازک امن نگاه‌دارد. پس این‌که به نظر شما کدام یک امن‌تر است؛ بسیار بدیهی است.

حال باید به دلیل آن که اطلاعات طراحی گاوصندوق اول در اختیار است و می‌توان به راحتی با تجهیزات و مواد اولیه لازم همانند آن طراحی کرد، گفت که شفافیت، باعث ناامن بودن گزینه اول خواهد بود؟ خیر، امنیت اطلاعات شما در گاو صندوق اول به دو دلیل است. یکی استفاده از ترکیبات و قدرت بالای آن در حفاظت اشیاء و ترکیبات امنیتی فوق‌العاده بالا است و دیگری به خاطر در دسترس بودن اطلاعات گاوصندوق برای کارشناسان که منجر به این می‌شود که آنان با کمک به طراحی و ارائه راهکارهایی جدید همیشه آن را بهتر از قبل نگاه می‌دارند. این گزینه باعث می‌شود که راهی مخفی در گاوصندوق برای گشودن آن وجود نداشته باشد. مثلا اگر مشکلی یا ایرادی در گاوصندوق وجود داشت، که با اطلاع از آن ساق می‌توانست آن را به سرعت باز کند، فردی خبره آن را متوجه شده و راهکار آن را ارائه خواهد کرد؛ زیرا که روش طراحی آن برای افراد در دسترس است.

هدف ما از امنیت چیست؟

چیزی نیست که کامل مطلق باشد با این حال تبلیغات در حال اغراق هستند، به عنوان مثال هیچ روش امنیتی وجود ندارد تا اجازه دزدی از اشیاء سرقت شده را به سارقان ندهد و یا کاملا آنان را متوقف کند، بلکه آن‌ها فقط قادرند هزینه دزدی از شما را برای سارقین بالا برند و سرقت از شما هزینه‌ای گزاف داشته باشد. این که بدانید آن روش امنیتی چگونه کار می‌کند آن را غیر امن نمی‌سازد. ممکن است علم به مواد اولیه و نوع فلز و یا فولاد به کار رفته در گاوصندوق باعث شود سارقین از ابزار مناسب به آن بهره جویند ولی چنین چیزی نیز باعث نمی‌شود که بگوییم چنین روش‌هایی برای سارقین کم‌هزینه باشند. گاوصندوقی با طراحی و ترکبات رمزی سخت می‌تواند هر مهاجمی را دلسرد کند.

قیاس ما از گاوصندوق و مکان امن، به راحتی می‌تواند موضوع نرم‌افزارهای امنیتی متن‌باز/آزاد را برای شما روشن کند.

همانند امنیت گاوصندوق مذکور امنیت نرم‌افزار قوی رمزنگاری به دلیل آزاد بودن برای شما پرخطر نیست. در واقع کد منبع نرم‌افزار امنیتی با دیده شدن توسط دیگران تقویت شده و بر حسب نیاز امنیت و حریم خصوص کاربران افزایش خواهد یافت. این همان دلیلی است که مرتبا اشاره می‌شود که نرم‌افزارهای آزاد با بررسی توسط کاربر نهایی ، می‌تواند از راه‌های پنهان و درهای پشتی عاری باشد. در دنیایی که نظارت بر کاربران در حال تبدیل به یک هنجار است؛ بدیهی است کاربران خواهان شفافیت در عرضه محصولات امنیتی باشند و لزوم حفاظت از حریم خصوصی و بالا بردن امنیت بسیار حیاتی است. این موضوع برای فعالین حقوق بشر، روزنامه‌نگاران، فعالین گروه‌های مدنی و اجتماعی «NGOها» و دیگران بازیگران عرصه عدل و عدالت که امنیت و حریم خصوصی برای آنان از نزدیک ملموس است، ضروری است.

شاید موضوع متن‌باز بودن کد منبع برنامه‌ای امنیتی منجر به بالا رفتن امنیت برنامه می‌شود را یک پارادوکس بدانید، با این حال با توجه به قیاس مذکور در ابتدای مقاله نرم‌افزاری که به صورت متن بسته عرضه می‌شود امنیت را بالاتر نخواهد برد؛ زیرا که امنیت به نحوه کار ترکیبات رمزی، الگوریتم‌های مورد استفاده و دیگر موارد موثر وابسته است. با این حال همانطور که در قیاس ذکر شد مهاجمان با مطالعه عمیق کدهای برنامه متن‌باز می‌توانند به مواردی دست‌یابند که باعث حمله به امنیت کاربران شود ولی امنیت به معنای عدم حمله و نبود ضعف نیست بلکه باعث پرهزینه شدن حمله و سرقت می‌شود.

ما در بنه‌تک بر این عقیده هستیم که استفاده از مدل متن‌باز برای توسعه به جامعه کاربران کمک خواهد کرد تا ما را در بهبود کدهایمان یاری دهند. در رابطه با موضوع مارتوس ما دیگر به اختراع مجدد چرخ نیازی نداشتیم و می‌توانستیم به راحتی از ابزارهای موجود مانند ابزار تور[2] «Tor» برای ناشناس ماندن و دیگر ابزار متن‌باز و آزاد بهره جوییم.

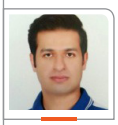
موضوع نرم‌افزارهای امنیتی برای فعالین حقوق بشر به موضوعی مهم برای فعالین تبدیل شده است که به طور روزافزونی خواهان پشتیبانی از فعالین امنیتی در این بخشها هستند. برخی از آنان بمانند بنیاد فن‌آوری باز که کاربران را برای بررسی کدها تشویق می‌نماید. به طور جامع استفاده از نرم‌افزارهایی با شفافیت بالا، مسیولیت‌پزیری بیشتر و اثبات‌پزیری قویتر منجر به بهبود امنیت نرم‌افزار خواهد شد که تمامی این موارد در نرم‌افزارهای آزاد قابل دسترسی هستند.■

۱. Benetech Human Rights Program
۲. Tor برنامه‌ای برای ناشناس ماندن در اینترنت است که از شبکه‌ای از کاربران در اینترنت شکل گرفته است.

پرونده

آشنایی با مدل‌های مدیریت داده‌ها

در دهه‌های گذشته، ذخیره‌سازی موثر داده‌ها و بدون تناقض آن‌ها جنبه مهم مدیریت داده‌ها بوده است. امروزه مساله ذخیره‌سازی داده‌ها سهم کمتری در میان مساله مدیریت داده‌ها دارد. مدیریت داده‌ها و استفاده از آن‌ها حتی نسبت به ده سال قبل، شکل دیگری به خود گرفته است. شبکه‌های اجتماعی، سیستم‌های امنیتی و هوشمند، میزبان‌های رسانه (Media Servers)، دوربین‌های مدار بسته، سیستم‌های مدیریت نقشه‌ها، شبکه‌های سنسوری و... هر یک به نوعی خاص نیازمند استفاده و مدیریت داده‌ها هستند. به عبارتی هر جنبه کیفی استفاده از داده‌ها مانند امنیت و سرعت دسترسی، امکان تغییر و توسعه داده‌ها به شکل‌های گوناگون و... باعث شدند مدل‌های ذخیره‌سازی متفاوتی برای داده‌ها به وجود آید. علاوه بر این موارد، حجم عظیم داده‌های مبادله شده باعث ایجاد مساله‌ای تازه به نام داده‌های بزرگ (BigData) شده است. بنابراین حوزه‌ی مطالعاتی در زمینه‌ی داده‌ها بسیار وسیع است. در این بخش سعی شده که به معرفی اجمالی از چند مدل ذخیره‌سازی داده‌ها و سیستم‌های مدیریت مربوط به آن مدل‌ها پرداخته شود. از آن‌جا که مدل داده‌ای یک پایگاه‌داده‌ها، رابطه مستقیمی با سیستم مدیریت آن پایگاه داده دارد (البته به جز نوع بدون قالب آن)، در این نوشته گاهی اوقات از واژه «مدل» و «پایگاه‌داده‌ها» به جای «سیستم مدیریت پایگاه‌داده‌ها» استفاده شده است. هرچند که هر کدام مفاهیمی جداهستند.



نویسنده: مهدی حمیدی

سیستم‌های مدیریت پایگاه‌های داده رابطه‌ای (RDBMS):

این مدل را می‌توان جزو پرکاربردترین و شناخته شده‌ترین مدل مدیریت داده‌ها و نیز بالغ‌ترین مدل بین سایر مدل‌ها دانست. شالوده مدل رابطه‌ای، منطق رابطه‌ای است. این منطق توسط ادگار کاد (Edgar Codd)، دانشمند علوم رایانه در سال ۱۹۷۰ معرفی شده و در واقع تفسیر دیگری از نظریه مجموعه‌ها و جبر رابطه‌ای است. هر قلم داده (Data Entity) در مدل رابطه‌ای، معادل یک سطر از جدول است که با یک کلید از سایر سطرهای آن جدول متمایز می‌شود. منطق رابطه‌ای در واقع یک جبر بسته است. به این ترتیب که همه چیز، اعم از جداول و سطرهای جدول، یک رابطه است و حاصل عملگرهای جبری بر روی رابطه‌ها، باز هم یک رابطه است. به خاطر این شالوده قدرتمند، مدل رابطه‌ای توانست جایگزین مدل‌های ناکارآمدی چون مدل سلسله مراتبی و مدل شبکه‌ای شود که پیمایش بین داده‌ها از طریق اشاره‌گرها و زیرروال‌های پیمایشگر صورت می‌گرفت؛ بر خلاف مدل رابطه‌ای که هر سطر به طور منظم و مستقل در جایگاه خود قرار دارد و یافتن آن‌ها نیازمند اشاره‌گرهای نسبی و فراخوانی زیرروال‌های پیمایشگر نیست. بسته بودن منطق رابطه‌ای، باعث به کارگیری پرس‌وجوها (query) به طور تو در تو و به طور موثر می‌شود. در این مدل، عملگرهای رابطه‌ای مانند join، باعث اتصال جدول‌ها با یکدیگر و استفاده‌ی موثر از داده‌ها می‌شود. زبان دستکاری و به کار گیری داده‌ها (DML) در این مدل، معمولا زبان SQL است که یک زبان سطح بالا و اخباری (declarative) است. از آن‌جا که روند تدریجی توسعه سیستم‌های نرم‌افزاری می‌تواند باعث تغییرات آتی در ساختار پایگاه‌داده‌ها شود، طراحی درست یک پایگاه‌داده‌ها، مساله‌ای ضروری و مهم است. این موضوع در مدل ذخیره‌سازی رابطه‌ای، تا حدودی مساله‌ای چالشی است. گرچه معرفی سطوح نرمال‌سازی و روش‌های نرمال‌سازی پایگاه‌های داده، گام‌های مورد نیاز را برای این امر فراهم می‌آورند، اما طراحی درست و نرمال یک پایگاه‌داده‌های رابطه‌ای خبرگی و مهارت خاص خود را می‌طلبد.

هر دستور یا مجموعه دستورات وابسته به هم که از طریق کاربران به پایگاه‌داده ارسال می‌شود، تحت پوشش یک تراکنش اجرا می‌شود. مدیریت تراکنش‌ها در این نوع شیوه ذخیره‌سازی، مساله‌ای جدا از خود مدل ذخیره‌سازی است؛ به عبارتی سیستم‌های مدیریت پایگاه‌های داده رابطه‌ای (RDBMS) به طور معمول موظفند در زمان اجرای تراکنش‌ها، خواص ACID را به نوعی رعایت و پیاده‌سازی کنند. بدین منظور هر یک از این سیستم‌ها شیوه و سیاست خاص خود را برای مدیریت تراکنش‌ها و پیاده‌سازی خواص ACID عرضه کرده‌اند. سیستم‌هایی مانند MySQL از شیوه قفل‌گذاری در سطح سطرهای جدول به این هدف می‌رسند. اما در سیستمی مثل PostgreSQL، این امکان توسط چند نسخه‌سازی صورت می‌گیرد.

مدل رابطه‌ای دارای محدودیت‌های خاص خود است. کاربر نمی‌تواند به طور معمول و موثر، هر نوع ساختار داده دل‌خواه خود را در یک جدول پایگاه‌داده‌های رابطه‌ای ذخیره کند. به علاوه این که هر RDBMS برای خود مجموعه نوع داده (Data Type)های خود را ارائه کرده است که جدای از ناهمخوانی با یکدیگر، محدود هستند

با معرفی منطق برنامه‌نویسی شئ‌گرا و زبان‌های مرتبط با آن و همه‌گیر شدن استفاده از این منطق در طراحی سیستم‌های نرم‌افزاری، این مشکل بیشتر خود را نمایش داد و یک شکاف بین منطق رابطه‌ای و منطق شیئ‌گرا حس شد. برنامه‌نویسان گاهی مجبور بودند همان‌طور که اشیاء داده را در برنامه‌شان استفاده می‌کردند، آن‌ها را در پایگاه‌های داده ذخیره و استفاده‌ی مجدد کنند و به این ترتیب زمان و هزینه توسعه را کاهش دهند. در ادامه با مدل‌هایی که سعی کردند این شکاف را برطرف کنند آشنا می‌شوید.

سیستم‌های مدیریت پایگاه‌های داده شئ گرا (OODBMS):

اواخر دهه ۸۰ میلادی را می‌توان زمان ظهور پایگاه‌های داده‌ی شئ‌گرا دانست که در واقع پاسخی به نیازمندی‌های برنامه‌های CAD بود که با اشیاء داده‌ی پیچیده و تودرتو سروکار دارند. مساله اینجاست که برنامه‌ی CAD یک برنامه‌ی روالی (procedural) است که در هر لحظه یک ورودی می‌گیرد و یک خروجی پیچیده تولید می‌کند، اما این با ساختار رابطه‌ای در تضاد است و پیاده‌سازی آن در منطق رابطه‌ای مستلزم اجرای دستورات join متعدد و پرس و جو (query)های طولانی، آن هم برای گرفتن تنها یک سطر خروجی است.

سیستم‌های مدیریت پایگاه‌داده‌های شئ‌گرا (OOD-BMS) برای حل این مشکلات به وجود آمد. خصوصیات این سیستم را می‌توان بطور خلاصه این طور بیان کرد:

- پشتیبانی از نوع داده کاربر (User Data Type)
- امکان ذخیره‌سازی اشیاء تودرتو
- پشتیبانی از لیست اشیاء (مانند مجموعه‌ها، آرایه‌ها، دسته‌ها) و bag (لیستی از لیست‌ها) ، هر کدام به عنوان یک شیئ مستقل
- پشتیبانی از متدهای اشیاء (معادل روال‌های ذخیره شده (Stored Procedure) در پایگاه‌داده‌های رابطه‌ای):

متدها جزئی از منطق شئ‌گرا هستند. به جای آن که برنامه درگیر اجرای query جهت دریافت اطلاعات اطلاعات از پایگاه‌داده‌های و محاسبات آن و ارسال نتیجه به پایگاه داده شود، یک متد شئ می‌تواند به طور موثر در سمت سرور پایگاه‌داده‌های اجرا شود و تغییرات را در همان جا، روی شئ اعمال کرده و نتیجه را به برنامه برگرداند.

و یکی از مهم‌ترین خصوصیات این سیستم، اشتراک در نوع داده در برنامه و پایگاه‌داده‌ها است که باعث اعمال قوانین بررسی قوی نوع (Strong Type Checking) در زمان انتقال داده‌ها بین برنامه و پایگاه‌داده‌ها، به طور موثر می‌شود.

ادغام با زبان‌های برنامه‌نویسی: از آن‌جا که ساختار داده‌ها در پایگاه‌داده‌ها، مانند همان ساختار برنامه کاربر است، ذخیره و بازیابی اطلاعات می‌تواند از طریق دستورات DML درون خود برنامه یا زبان برنامه نویسی صورت بگیرد، به جای آن‌که دستورات DML از طریق یک واسطه محدود مانند Connection Stringها یا دستورات خاص، به پایگاه‌داده‌ها، به عنوان یک سیستم جدا ارسال شود.

مشکلات سیستم‌های مدیریت پایگاه‌های داده شئ گرا:

پیمایش بین داده‌ها از طریق روال‌های جداگانه: در منطق پایگاه داده شیئ‌گرا، اشیاء داده به وسیله‌ی اشاره‌گرها به یکدیگر متصل هستند. این منطق وجود یک زیرروال پیمایشگر را موجب می‌شود که می‌توان آن را یک گام به عقب توصیف کرد.

نقض encapsulation: خاصیت encapsulation یکی از پایه‌های مفهوم شیئ‌گرایی است. از آن‌جا که با اجرای query می‌توان به داده‌های درون یک شئ درون پایگاه‌داده‌ها دست یافت، می‌توان این طور گفت که پایگاه‌داده‌ها شئ‌گرا، مفهوم شئ‌گرایی را به طور کامل پشتیبانی نمی‌کنند. گاهی اوقات نقض این encapsulation ضروری است. به عنوان مثال نیازمند داده‌ای از یک شئ داده هستیم که متد آن پیاده‌سازی نشده است و دستیابی به این مقدار، تنها یک بار صورت می‌گیرد. در واقع منطقی نیست که برای دستیابی به این مقدار بخواهیم یک متد جدا درون شئ داده‌ی درون پایگاه‌داده‌ها بنویسیم. بنابراین همواره یک مصالحه بین نقض encapsulation و نوشتن متدهای بیشتر برای اشیاء داده وجود دارد.

بسته بودن جبر رابطه‌ای موجود در منطق رابطه‌ای در این‌جا وجود ندارد و به کارگیری پرس‌وجو های تودرتو که حاصل این بسته بودن بود، در این‌جا از دست می‌رود. به خصوص که در این مدل، اخباری بودن و سطح بالا بودن زبان دسترسی به داده‌ها (DML) تضعیف می‌شود.

در نهایت آن استخوان‌بندی و شالوده قدرتمند ریاضی که منطق رابطه‌ای بر آن استوار بود، در منطق پایگاه‌داده‌های شئ‌گرا وجود ندارد و این به معنی از دست رفتن مجموعه‌ای از ابزارهای تحلیل و استنتاج مبتنی بر این منطق است.

ایده‌ای که در پشت پیاده‌سازی پایگاه‌داده‌های شئ گرا نهفته بود این بود که برای توسعه‌دهندگان بسیار مفید خواهد بود که نه تنها ذهنشان درگیر پیاده‌سازی پشت رابط اشیاء داده نشود، بلکه حتی درگیر این مساله نباشند که آن شئ چطور در پایگاه‌داده‌ها ذخیره می‌شود.

سه اشتباه که تکرار شد:

شرکت‌های توسعه‌دهنده و پشتیبان ایده پایگاه‌های داده شئ‌گرا متوجه تکرار اشتباه خود شدند.

اولین اشتباه آن‌ها نزدیک کردن بیش از حد طراحی پایگاه‌داده‌ها به طراحی برنامه‌ها بود. آن‌ها دوباره فهمیدند که این نزدیک شدن با معماری چند لایه توسعه نرم‌افزارها که باعث انعطاف در توسعه نرم‌افزارها می‌شود تناقض دارد.

آن‌ها همچنین دوباره فهمیدند که استفاده از زبان‌های اخباری مانند SQL-۹۲ چنان بهره‌وری را بالا می‌برد که شرکت‌های توسعه‌دهنده نرم‌افزار ترجیح می‌دهند که به جای درگیری با مفاهیم جدید، هزینه تامین منابع سخت‌افزاری را پرداخت کنند؛ به عبارتی شما همواره می‌توانید سخت‌افزار بخرید، اما زمان را هرگز!

آن‌ها همچنین دوباره فهمیدند که استاندارد نبودن یک مدل داده‌ای منجر به خطا در طراحی و بروز ناسازگاری در داده‌ها می‌شود. نگهداری از یک سیستم مدیریت پایگاه‌های داده شئ‌گرا کاری بسیار سخت بود.

سیستم مدیریت پایگاه‌های داده شئ-رابطه‌ای (ORDBMS):

گرچه تا سال ۹۵ میلادی، بحث شئ‌گرایی در پایگاه‌های داده مساله داغی بود، اما هنوز یک توافق عام بر سر این‌که یک پایگاه‌داده‌های شئ‌گرا باید شامل چه خصوصیاتی باشد، شکل نگرفت! در نهایت همگان بر این موضوع توافق کردند که پشتیبانی کمی شئ‌گرایی در پایگاه‌های داده بسیار مفید است! به عبارتی گرچه پایگاه‌های داده شئ‌گرا نتوانستند موفقیت قابل انتظاری را کسب کنند، اما پیشرو در ارائه چند ایده موفق و کاربردی بودند که بعدها در پایگاه‌داده‌های شئ-رابطه‌ای نیز توانستند پیاده‌سازی شوند. همچنین توسعه‌دهندگان سیستم‌های مدیریت پایگاه‌داده‌های رابطه‌ای نیز متوجه وجود کمبود در این سیستم‌ها شدند، سعی کردند به هر طریقی پشتیبانی از اشیاء داده را در سیستم‌های خود و با رعایت انطباق با نسخه‌های قبلی وارد کنند. که در منبع چهارم این نوشته با دشواری‌های زیاد این رویکرد آشنا خواهید شد. نهایتا گرایش به فواید موجود در مدل‌های ORD-BMS و OODBMS باعث به وجود آمدن مفهومی با عنوان پایگاه‌های داده شئ-رابطه‌ای (Object-Relational) شد.

در این مفهوم به طور موثری تلاش شده که نه تنها نقات قوت هر دو سیستم پیاده‌سازی شده و نقاط ضعف آن‌ها برچیده شود، بلکه این سیستم بیانگر یک راهکار نوین در زمینه مدیریت داده‌ها محسوب شود. سیستم مدیریت پایگاه‌داده‌های که در این زمینه پیشرو بوده است، PostgreSQL است. PostgreSQL را می‌توان یک سیستم پایگاه‌داده‌های شئ-رابطه‌ای بسیار قدرتمند و انعطاف‌پذیر دانست که علاوه بر معرفی راهکاری در جهت پشتیبانی از اشیاء داده‌ها، استانداردهای روز پایگاه‌داده‌های رابطه‌ای را پیاده‌سازی می‌کند. مشخصات کلی یک سیستم شئ-رابطه‌ای را می‌توان به این صورت بیان کرد:

پشتیبانی از داده‌های پیچیده

پشتیبانی از ارث‌بری نوع داده

پشتیبانی از رفتار اشیاء

منظور از رفتار اشیاء همان متدهای تعریف شده درون اشیاء داده است و این رفتار را می‌توان به عنوان حتی یک برنامه‌ی مستقل گسترش داد. نکته جذاب این است که با این دیدگاه می‌توان بخش زیادی از لایه منطق (Business Logic) برنامه را در سمت سرور اجرا کرد، بدون آن‌که مانند گذشته، محاسبات در سمت کلاینت صورت بگیرد و تاخیر در انتقال داده‌ها باعث تاخیر در محاسبات گردد. یک سیستم مدیریت پایگاه‌داده‌های شئ-رابطه‌ای مانند یک سکو یا با یک مثال بد، مانند یک سیستم‌عامل عمل می‌کند، به این ترتیب که می‌تواند منابع سیستم را برای این برنامه‌ها و نیز ارتباطات بین برنامه‌ای را کنترل و مدیریت کند. این قدرت و انعطاف‌پذیری، باعث شده است که بسته‌های مختلفی برای این نوع سیستم نوشته شود که نوع داده‌ها و متدهای خود را وارد ORDBMS می‌کنند و کاربر می‌تواند در کنار استفاده‌ای که قبلا از یک DBMS به عنوان یک سیستم رابطه‌ای استاندارد می‌کرد، از این متدها و نوع داده‌های جدید در برنامه خود استفاده نماید.

در این نوع پایگاه‌داده‌های، پشتیبانی از شئ‌گرایی به طور موثری از طریق جداول مسطح صورت می‌پذیرد و پشتیبانی از نوع داده‌هایی مانند کلکسیون‌ها، خللی در مفهوم نرمال بودن پایگاه‌داده‌های از دید رابطه‌ای آن وارد نمی‌کند.

معرفی سیستم‌های مدیریت پایگاه‌داده‌های NoSQL و مخازن داده:

گرچه مدل رابطه‌ای بسیار قدرتمند و انعطاف‌پذیر است، اما همان‌طور که اشاره شد، کارکردن با این مدل داده مهارت خاص خود را می‌طلبد. جدای از این، خیلی از کاربران، مشکلات و نیازمندی‌هایی دارند که راهکار پایگاه‌داده‌های رابطه‌ای و حتی شیء-رابطه‌ای برایشان مناسب نیست. در دهه اخیر سیستم‌هایی موسوم به NoSQL و برنامه‌های مربوط به آن‌ها بسیار پرطرفدار و فراگیر شده‌اند، با این شعار که در کنار معرفی کارکردهای جدید، بسیاری از دشواری‌های کار کردن با مدل رابطه‌ای را نیز برطرف کنند. در واقع هدف این است که با ریشه‌کن کردن محدودیت‌های سخت‌گیرانه‌ای که قبلا در سیستم رابطه‌ای معرفی شده بود، بتوان به راحتی و انعطاف بالا در نگهداری، پرس‌وجو و استفاده از داده‌ها رسید. پایگاه‌داده‌های NoSQL با استفاده از شیوه‌های ساخت‌نایافته (یا ساخت‌یافته در زمان عمل) به حذف محدودیت‌های رابطه‌ها کمک می‌کند و راه‌های متعدد و مختلفی را جهت موارد استفاده‌ی خاصی هم چون ذخیره «تمام متن» اسناد (full-text document storage) ارائه می‌کند. بر خلاف آن‌چه که در مقدمه آورده شده، در سیستم‌های پایگاه‌داده‌های NoSQL هیچ مدل داده‌ای همانند سیستم‌های پایگاه‌داده‌های رابطه‌ای، مورد استفاده یا نیاز نیست. هر سیستم مدیریت پایگاه‌داده‌های NoSQL، به منظورهای خاص و هر یک به شیوه خاص، این منطق را پیاده سازی کرده‌اند. این پیاده‌سازی‌ها و راهکارهای بدون قالب (schema less) هم اجازه‌ی پشتیبانی از انواع نامتناهی از فرم‌های داده‌ای را فراهم می‌آورد و هم این‌که به طور ساده و بسیار موثری، از قالب «کلید-مقدار»ی که در مدل رابطه‌ای بود، پشتیبانی می‌کنند. پشتیبانی از قالب کلید-مقدار، بیشتر برای تعداد داده‌های کوچک و معمولا به منظور cache کردن داده‌ها به کار می‌رود که در بخش مقایسه سیستم‌های NoSQL به آن پرداخته می‌شود.

بر خلاف پایگاه‌داده‌های رابطه‌ای، قادر خواهیم بود تا کلکسیونی (collection) از داده‌ها را درون یک سیستم پایگاه‌داده‌های NoSQL همانند MongoDB داشته باشیم. این پایگاه‌داده‌های یا به عبارت دیگر این «مخازن اسناد» (document stores) هر داده را در کنار سایر داده‌ها، تحت عنوان یک کلکسیون (یا همان سند) در پایگاه‌داده‌های نگهداری می‌کند. این اسناد می‌توانند تحت عنوان اشیاء داده مستقل مانند قالب JSON به نمایش درآیند و همچنین بر اساس صفت‌هایش پرس‌وجو شوند.

سیستم‌های پایگاه‌داده‌های NoSQL بر خلاف سیستم رابطه‌ای که از زبان SQL استفاده می‌کرد، روش‌های مشترکی را برای پرس‌وجو از داده‌ها ارائه نمی‌کنند و هر سیستم NoSQL راهکار پرس‌وجو یا دسترسی خاص خود به داده‌ها را دارا است.

مقایسه سیستم‌های مدیریت پایگاه‌های داده مبتنی بر SQL (رابطه‌ای) و NoSQL

جهت ساده‌سازی در ارائه‌ی مفهوم، سعی می‌گردد تا تفاوت‌های این دو سیستم بررسی گردد

- ساختار و نوع داده نگهداری شونده: در سیستم رابطه‌ای، نیازمند یک ساختار از صفات برای نگهداری از داده‌ها هستیم، بر خلاف NoSQL که معمولا اجباری را تحمیل نمی‌کند.
- پرس‌وجو: همه پایگاه‌داده‌های رابطه‌ای استاندارد SQL را تا حدودی پیاده‌سازی کرده‌اند و می‌توانند از طریق SQL مورد پرسوجو قرار گیرند. در NoSQL شرایط عکس است. همان طور که گفته شد، هر پیاده‌سازی، روش خودش را برای کار با داده‌ها دارد.
- بسط‌پذیری: هر دو این راهکارها قابل رشد به صورت عمودی هستند. (مثلا افزایش منابع سیستم). هرچند با پیشرفت و ساده‌تر شدن برنامه‌ها، راهکارهای NoSQL ابزارهای راحت‌تری را در جهت رشد افقی ارائه می‌کنند (مثلا ساختن یک cluster از چندین ماشین)
- قابلیت اتکا: هنگامی که مساله‌ی قابلیت اطمینان در داده‌ها و تضمین تراکنش‌های انجام شده مطرح باشد، همچنان بهتر است بر روی پایگاه‌های داده‌های رابطه‌ای شرط ببندید!
- پشتیبانی: پایگاه‌داده‌های رابطه‌ای، قدمتی طولانی دارند. همان طور که گفته شد، بسیار پرطرفدار هستند و پیدا کردن پشتیبانی برای آن‌ها چه به صورت رایگان و هزینه‌ای به راحتی امکان‌پذیر است. طبیعی است که حل مشکلات نیز در سیستم سنتی بسیار سریع‌تر از سیستم‌های تازه به ظهور رسیده مانند MongoDB که گفته می‌شود ذاتا پیچیده است، خواهد بود.
- داده‌های پیچیده و نیاز به نگهداری و پرس‌وجو: به طور ذاتی، پایگاه‌داده‌های رابطه‌ای از منطق go to جهت پرس‌وجو های پیچیده و برآورده نمودن نیاز نگهداری از داده‌ها استفاده می‌کنند که در این حوزه پیشرو بوده و بسیار موثرترند.
- گرچه در ادامه به جزئیات بیشتری از سیستم‌های NoSQL پرداخته می‌شود، لیکن موارد استفاده یا عدم استفاده از این سیستم‌ها را می‌توان به طور کلی به این شکل عنوان کرد:
- اندازه کار: اگر با دسته‌های عظیم از داده‌ها سروکار دارید، بسط دادن این داده‌ها توسط خانواده پایگاه‌داده‌های NoSQL راحت‌تر صورت می‌گیرد.
- سرعت: پایگاه‌داده‌های NoSQL معمولا سریعترند، و بعضی وقت‌ها در زمان عملیات نوشتن به شدت سریعترند! عملیات خواندن هم بسته به نوع پایگاه داده NoSQL و داده‌های مورد پرس‌وجو می‌تواند بسیار سریع صورت بگیرد.
- طراحی بدون قالب: همان طور که گفته شد، پایگاه‌داده‌های NoSQL قابلیت انعطاف بالایی دارند.
- بسط‌پذیری و تکرارپذیری آسان: بر خلاف پایگاه‌داده‌های رابطه‌ای، پایگاه‌داده‌های NoSQL به راحتی قابل بسط و تکرار هستند
- آزادی عمل در انتخاب: تنوع در پیاده‌سازی‌های مختلف از مفهوم NoSQL، این امکان را به کاربران می‌دهد تا با توجه به مساله و داده‌ای که با آن کار می‌کنند، سیستم مدیریت پایگاه‌داده‌های مناسب را انتخاب کنند. در ادامه با این موارد قابل انتخاب آشنا می‌شوید.

مقایسه سیستم‌های مدیریت پایگاه‌داده‌های NoSQL و مدل‌های آن‌ها

در این بخش سعی بر آن است که سیستم‌های مدیریت پایگاه‌داده‌های NoSQL معرفی گشته و کارکرد و اهداف آن‌ها تشریح شود تا بتوان در صورت نیاز، انتخابی درست و موثر انجام داد.

به طور کلی، می‌توان سیستم‌های مدیریت پایگاه‌های داده NoSQL را به چهار دسته زیر تقسیم نمود:

- مبتنی بر مقدار-کلید
- مبتنی بر ستون
- مبتنی بر سند
- مبتنی بر گراف

1) مبتنی بر کلید-مقدار:

این مدل می‌تواند به عنوان ساده ترین مدل در بین سایر مدل‌های این دسته و نیز زیرساخت مفهوم NoSQL شناخته شود. این نوع از پایگاه‌داده‌های با تطابق کلیدها و مقادیر کار می‌کنند، چیزی شبیه dictionary. هیچ چیزی به عنوان ساختار یا رابطه وجود ندارد. پس از اتصال به پایگاه داده (مانند Re-dis)، یک برنامه می‌تواند یک کلید را عنوان کند (مانند the_answer_to_life) و یک مقدار متناسب (مثلا ۴۲) را دریافت کند. سیستم‌های مدیریت پایگاه‌داده‌های مبتنی بر مقدار-کلید بطور معمول برای ذخیره‌ی سریع داده‌های اولیه به کار می‌روند. آن‌ها بسیار کارا، موثر و معمولا بسط پذیرند.

نکته: وقتی صحبت از دیکشنری در دنیای کامپیوتر

می‌شود، منظور دسته‌ای خاص از اشیاء داده است که از آرایه‌ای از کلکسیون‌ها با کلیدهای مجزا تشکیل شده اند و هر کلید، با یک مقدار تطابق دارد.

برای سیستم‌های مدیریت پایگاه‌داده‌های NoSQL مبتنی بر کلید-مقدار می‌توان موارد زیر را معرفی نمود:

- Redis: درون حافظه اصلی قرار می‌گیرد و امکان ماندگار بودن در حافظه را نیز دارد.
- Riak: قابلیت بسط‌پذیری بالا، و قابلیت چند نسخه‌سازی
- Memcached / MemcacheDB: قابلیت بسط‌پذیری

موارد مناسب جهت استفاده:

- caching: سرعت ذخیره بالا، و گاهی تکراری داده‌ها و با هدف استفاده در آینده نزدیک
- در صف قرار دادن: برخی از سیستم‌ها مانند Redis از لیست‌ها، دسته‌ها و صف‌ها و... پشتیبانی می‌کنند.
- توزیع عملیات و اطلاعات: می‌توانند جهت پیاده‌سازی الگوی معماری Publish-Subscribe موثر باشند
- نگهداری از اطلاعات زنده: برنامه‌هایی که نیازمند نگهداری از «حالت» (state) هستند، می‌توانند از این سیستم‌ها استفاده کنند.

۲) مبتنی بر ستون:

سیستم‌های مدیریت پایگاه‌های داده مبتنی بر ستون، بر اساس قدرتمندسازی همان طبیعت کلید-مقدار شکل گرفته‌اند. بر خلاف آن تصور عامه در اینترنت که گفته می‌شود یادگیری این پایگاه‌داده‌ها مشکل است، این نوع پایگاه‌داده‌ها به طور خیلی ساده و بر اساس ساخت کلکسون‌هایی از یک یا چند جفت کلید-مقدار کار می‌کنند که با یک رکورد مطابقت دارد. بر خلاف تعاریف قدیمی از قالب (schema) در سیستم‌های رابطه‌ای، یک سیستم مبتنی بر ستون نیازمند جدولی از پیش تعیین شده جهت کار کردن با داده‌ها در آن نیست. هر رکورد با یک یا چند ستون که دربردارنده اطلاعات هستند، ارائه می‌شود و هر ستون از هر رکورد می‌تواند متفاوت باشد.

به طور ساده، این نوع پایگاه‌داده‌ها، آرایه‌هایی دو بعدی هستند که به موجب آن، هر کلید (در اینجا: معادل سطر یا رکورد)، یک یا چند جفت کلید-مقدار را به همراه دارند که به آن‌ها متصل هستند. این سیستم، اجازه استفاده و نگهداری حجم عظیمی از داده‌های ساخت‌نایافته را می‌دهد. (مثلا یک رکورد با مقادیر عظیمی از اطلاعات)

این سیستم معمولا زمانی استفاده می‌شود که جفت‌های کلید-مقدار کافی نیستند و ذخیره تعداد بسیار زیاد از رکوردها با تعداد اطلاعات بسیار بزرگ، یک ضرورت محسوب می‌شود. این سیستم قابلیت بسط‌پذیری بسیار خوب و بالایی را دارد.

برای سیستم‌های مدیریت پایگاه‌داده‌های مبتنی بر ستون می‌توان موارد زیر را معرفی نمود:

- Cassandra: مخزن داده مبتنی بر BigTable و Dynamo-DB
- HBase: مخزن داده سکوی Apache Hadoop، مبتنی بر ایده‌ی BigTable

موارد مناسب جهت استفاده:

- نگهداری از داده‌های ساخت‌نایافته و اطلاعات غیرفرار: اگر دسته‌ای عظیم از مقادیر و صفات نیازمند به نگهداری برای مدت طولانی باشند، این سیستم بسیار کاربردی است.
- بسط‌دهی: این سیستم‌ها به طور ذاتی قابلیت بسط پذیری بسیار بالایی دارند و می‌توانند با مقادیر بسیار بسیار زیادی از اطلاعات سروکار داشته باشند.

۳) مبتنی بر سند:

سیستم مدیریت پایگاه‌داده‌های مبتنی بر سند را می‌توان آخرین دیوانه بازی دانست که توانسته سیل عظیمی از مردم را به استفاده از خود جوگیر کند! این سیستم همانند سیستم مبتنی بر ستون عمل می‌کند، گرچه امکان رسیدن به تودرتوسازی بیشتری را می‌دهد: وجود یک سند در یک سند دیگر، در سند دیگر..

سندها بر محدودیت یک یا دو سطح از تودرتو سازی سیستم‌های مبتنی بر ستون فائق آمده‌اند. به طور ساده، هر ساختار پیچیده و دلخواه می‌تواند تشکیل یک سند را بدهد و این سند قابلیت ذخیره‌سازی در این سیستم را دارد.

علیرغم ذات قدرتمند قابلیت پرس‌وجو از رکوردها با کلیدهای مختلف، این سیستم مشکلات و افت‌هایی را در مقایسه با بقیه همتایان خود دارد. به عنوان مثال، دریافت یک مقدار از یک رکورد به معنی دریافت مقدار زیاد آن رکورد است. مشابه این سناریو در زمان به‌روزرسانی داده‌ها نیز صادق است. این موارد باعث افت کارایی می‌شوند.

برای سیستم‌های مدیریت پایگاه‌های داده مبتنی بر سند می‌توان موارد زیر را معرفی نمود:

- MongoDB: پایگاه‌داده بسیار پرطرفدار و بسیار کارا
- CouchDB: یک مخزن داده‌ی پیشرو و سنت شکن
- Couchbase: مبتنی بر JSON، قابلیت ذخیره در حافظه‌اصلی

موارد مناسب جهت استفاده:

- اطلاعات تودرتو: این سیستم‌ها اجازه کار با داده‌های بسیار پیچیده و تودرتو را می‌دهد.
- کار با جاوااسکریپت: یکی از حیاتی‌ترین کارکردهای این نوع سیستم‌ها، روشی است که در آن با برنامه‌های جاوا اسکریپتی روبرو می‌شوند، مثلا استفاده از قالب JSON مناسب برای جاوا اسکریپت!

۴) مبتنی بر گراف:

در نهایت، می‌توان به سیستم جذاب مدیریت پایگاه‌داده‌های مبتنی بر گراف اشاره کرد. این سیستم‌ها داده‌ها را به شکلی کاملا متفاوت از سه مدل قبلی که به آن اشاره شد، ارائه می‌کنند. آن‌ها ساختار درختی را همانند آن‌چه که در گراف‌ها هست ارائه می‌کنند، با راس‌ها و یال‌هایی که به وسیله‌ی رابطه‌ها به یکدیگر متصل هستند.

مشابه تصویری ریاضی گراف‌ها و به لطف طبیعت گراف‌ها (مانند اتصال و دسته‌بندی اطلاعات مرتبط با هم)، برخی از عملیات خاص در این سیستم‌ها، بسیار ساده‌تر صورت می‌پذیرند. (مانند ارتباط آدم‌ها)

این پایگاه‌داده‌های به طور معمول در برنامه‌هایی استفاده می‌شوند که برقراری مرزهای مشخص در اتصالات ضروری است. به عنوان مثال هنگامی که وارد یک شبکه اجتماعی شویم، اتصال دوستانتان به شما و نیز اتصال دوستان دوستانتان به شما از طریق سیستم‌های مدیریت پایگاه‌داده‌های مبتنی بر گراف ساده‌تر است.

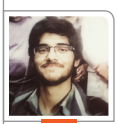
برای سیستم‌های مدیریت پایگاه‌داده‌های مبتنی بر گراف می‌توان موارد زیر را معرفی نمود:

- OrientDB: یک پایگاه‌داده‌های ترکیبی NoSQL از گراف و سند و بسیار سریع که با جاوا نوشته شده و قابلیت کار در مدل‌های کاری مختلف را دارد.
- Neo4J: پایگاه‌داده‌های بسیار پر طرفدار و قدرتمند مبتنی بر جاوا

موارد مناسب جهت استفاده:

- سروکار داشتن با داده‌های رابطه‌ای پیچیده: همان طور که گفته شد، این سیستم برای کار با ساختارهای پیچیده و رابطه‌ای بسیار موثر و راحت است. مانند ارتباط بین دو موجودیت و هر درجه از ارتباط بین موجودیت‌هایی که به طور غیر مستقیم با یک موجودیت رابطه دارند.
- مدل سازی و سروکار داشتن با موجودیت‌های دسته‌بندی شده: این سیستم‌ها در هر زمینه‌ای که مفهوم روابط انسانی مطرح است، برتری دارند. دسته‌بندی اطلاعات بر طبق روابط انسانی می‌توانند به طور خیلی خوبی توسط این سیستم‌های انبار داده صورت گیرند.

با انواع پایگاه‌داده‌های پرکاربرد آشنا شوید



نویسنده:
محمد خلاقی

MongoDB

نوع: NoSql

مونگودی‌بی یک پایگاه داده مستقل از بستر است که بر خلاف پایگاه‌داده‌های معمول که بر پایه جدول و پایگاه داده‌های رابطه‌ای (relation-al database) پیاده‌سازی می‌شوند، مبتنی بر سند (document-oriented) پیاده‌سازی شده و با استفاده از یک قالب پویای شبیه JSON به یک‌پارچه‌سازی داده‌ها می‌پردازد. مونگودی‌بی این قالب را BSON می‌خواند که در انواع خاصی از کاربردها، سریع‌تر از پایگاه‌داده‌های رابطه‌ای معمول عمل می‌کند.

لازم به ذکر است که مونگودی‌بی جزو پایگاه‌داده‌های NoSQL دسته‌بندی می‌شود.

مونگودی‌بی نخستین بار در ۲۰۰۷ توسط شرکت MongoDB inc ساخته و توسعه داده شد. این شرکت از سال ۲۰۰۹ به مدل توسعه آزاد روی آورد و از طریق پشتیبانی خدمات خود برای شرکت‌ها، کسب درآمد می‌کند

مونگودی‌بی با استفاده از زبان‌های برنامه‌نویسی سی، سی‌پلاس‌پلاس و جاوااسکریپت نوشته شده است.

تاکنون مونگودی‌بی توسط مراکز و شرکت‌های بزرگی چون لینکدن، سورس‌فورج، ای‌بی، فوراسکوئر، نیویورک تایمز و به خدمت گرفته شده است. به علاوه مونگودی‌بی معروف‌ترین پایگاه داده‌ها NoSQL محسوب می‌شود.

تفاوت اصلی مونگودی‌بی با پایگاه‌داده‌های معمول در نوع نگه‌داری داده‌ها و رابطه میان آن‌هاست. مونگودی‌بی به جای این که از جدول‌ها برای نگه‌داری داده‌های خود استفاده کند، از چندین سند ساده برای این کار استفاده می‌کند. برای مثال به جای این که اطلاعات نویسنده و عنوان و قیمت را درجدول‌های متمایز مربوط به هم نگه‌داری کنیم، در مونگودی‌بی همه اطلاعات در یک سند نگه‌داری می‌شود.

همچنین مونگو قابلیت بالایی در مجموعه‌های تکراری دارد، مجموعه تکراری مجموعه‌ای است که شامل بیش از دو نسخه از یک داده باشد. معمولا از این قابلیت به این صورت استفاده می‌شود که یک عضو اصلی، تمامی اعمال مربوط به خواندن و نوشتن را انجام می‌دهد و عضوهای دیگر، از عضو اصلی رونوشت تهیه می‌کنند. درصورتی که عضو اصلی در انجام کار خود ناکام بماند، دیگر عضوها طی یک فرآیند رأی‌گیری، یکی از عضوها را به عنوان عضو اصلی انتخاب می‌کنند تا عملیات عضو اصلی را انجام دهد.

مونگو از سیستم sharding برای پخش کردن بار پردازشی سامانه استفاده می‌کند که موجب می‌شود هنگام بروز خطای سخت‌افزاری، سامانه با مشکل مواجه نشده و به کار خود ادامه دهد.

از دیگر ویژگی‌های مونگو می‌توان به مجموعه‌های پوش اشاره کرد. این مجموعه‌ها اندازه‌های ثابتی دارند و زمانی که مجموعه به اندازه مورد نظر رسید، مانند یک فهرست دایره‌ای عمل می‌کند.

مونگو هم‌چنین از کاهش نگاشت پشتیبانی می‌کند.

مزایای استفاده از مونگو را می‌توان در موارد زیر خلاصه کرد:

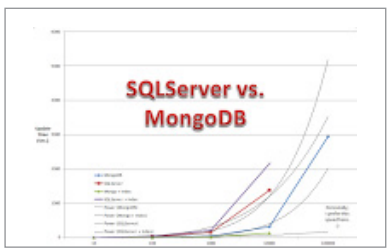
- استفاده از ساختار سندمحور که باعث می‌شود اطلاعات مربوط به یک نمونه، در سند آن و با استفاده از قالبی شبیه JSON ذخیره شوند.
- ساختار هر شی مشخص است.
- مونگودی‌بی از ساختارهای پیچیده به دور است.
- استفاده از حافظه داخلی دستگاه برای ذخیره‌سازی کارها که موجب دسترسی سریع‌تر به داده‌ها می‌شود.

معایب مونگودی‌بی

نمایه‌گذاری فیلدها به دلیل استفاده از درخت دودویی، موجب اشغال فضای نسبتا زیادی از حافظه سامانه می‌شود.

اندازه هر سند به دلیل تغییرپذیری احتمالی در آینده، از ابتدا بسیار بالا درنظر گرفته شده و در مواردی که یک سند با اطلاعات کم هم داشته باشیم، حجم زیادی فضا اشغال می‌شود.

- مونگو از تراکنش‌ها پشتیبانی نمی‌کند.
- در برخی موارد، انعطاف‌پذیری پرس‌وجو در مونگو از رقبای مبتنی بر Sql آن کم‌تر است.



MySQL

نوع: سامپادرا (RDBMS)

مای‌اس‌کیوال در رده سامپادرا یا سامانه مدیریت پایگاه داده رابطه‌ای قرار می‌گیرد. مای‌اس‌کیوال دومین سامپادرای بزرگ جهان است که بزرگ‌ترین سامپادرای آزاد جهان هم محسوب می‌شود.

مای‌اس‌کیوال با پروانه جامع همگانی گنو منتشر شده است. این پایگاه داده در ابتدا توسط یک شرکت سوئدی به نام MySQL AB ساخته و توسعه داده شد و اکنون اوراکل مالک آن به حساب می‌آید . نخستین نگارش مای‌اس‌کیوال در ۱۹۹۵ انتشار یافت.

مای‌اس‌کیوال محبوب‌ترین پایگاه داده برای استفاده

در وب است. برای مثال سامنه‌های مدیریت محتوای معروفی مانند جوملا، وردپرس، دروپال و... از آن استفاده می‌کنند. هم‌چنین خدمات بزرگ اینترنتی نظیر گوگل، فیس‌بوک، توییتر، فلیکر، یوتیوب و... نیز از مای‌اس‌کیوال به عنوان پایگاه داده خدمات خود استفاده می‌کنند.

مای‌اس‌کیوال با استفاده از سی و سی‌پلاس‌پلاس پیاده‌سازی شده و تجزیه‌گر آن با یَک نوشته شده است.

مای‌اس‌کیوال روی بسیاری از بن‌سازه‌های موجود مثل گنو/لینوکس، خانواده بی‌اس‌دی، مک اواس ده، ویندوز و... کارمی‌کند.

مزایای مای‌اس‌کیوال

فرآیند نصب و راه‌اندازی مای‌اس‌کیوال بسیار ساده است و نرم‌افزارهای سوم شخص بسیاری که برایش نوشته شده است، کار کردن با آن را چندین برابر آسان‌تر و سریع‌تر کرده است. همین‌طور استفاده از خود پایگاه داده بسیار ساده است.

در واقع اصلی‌ترین چالش کار با مای‌اس‌کیوال، یادگیری زبان SQL و موتورهای مختلف آن است.

مای‌اس‌کیوال یک استاندارد صنعتی و البته پایگاه داده بسیار محبوب است. این امر موجب گسترده بودن جامعه کاربری این سامانه شده است که پشتیبانی فنی قوی این محصول، ناشی از همین جامعه کاربری بزرگ است.

معایب مای‌اس‌کیوال

وجود برخی ناپایداری‌های جزئی در مای‌اس‌کیوال در برابر دیگر سامپادراها.

PostgreSQL

نوع: سامپادراش (ORDBMS)

پُستگرس‌کیوال پایگاه داده‌هایی از نوع سامپادراش محسوب می‌شود که هدف اصلی آن تطبیق با استانداردها و توسعه‌پذیری بالا است، این پایگاه داده‌ها مستقل از بستر بوده و روی همه سیستم‌عامل‌ها اجرا می‌شود. با این حال، بیشتر روی سیستم‌عامل‌های گنو/لینوکس و ویندوز استفاده می‌شود.

پستگرس، نخستین بار در دانشگاه برکلی و در ادامه پروژه اینگرس که یک سامپادرا بود در سال ۱۹۸۶ کلید خورد و اکنون توسط جامعه کاربری پستگرس که شامل خیل فراوانی از داوطلبان هستند، توسعه داده می‌شود. در سال ۱۹۹۴ اندرو یو و جولی چن، مفسر اس‌کیوال را به پستگرس اضافه کرده و آن را با نام پستگرس ۹۵ منتشر نمودند.

در مقایسه با سامپادراهای معمول، پستگرس از بُعد قابلیت، با نیاز عمیقش به شی‌گرائی، از مفهوم رابطه‌ای متمایز می‌شود.

بر اساس همین زیرساخت قوی است که پستگرس، توانایی خوبی در به دست گرفتن بهینه تعداد زیادی عملیات دارد.

از دیگر مزایای این پایگاه داده، می‌توان به این نکته اشاره کرد که پستگرس، به شدت توسعه‌پذیر بوده و با استفاده از روش‌های شخصی‌سازی، می‌توان از آن برای اجرای ساده عملیات پیچیده و تکراری استفاده کرد.

مزایای پستگرس‌کیوال

هم‌چون دیگر پایگاه‌داده‌های معروف، یکی از نقاط قوت پستگرس نیز جامعه کاربری بالای آن است و همین‌طور نرم‌افزارهای سوم‌شخصی که برایش منتشر شده و کاربری آن را آسان نموده‌اند.

توسعه پذیری

از ویژگی‌های مهم پستگرس می‌توان به توسعه‌پذیری بالای آن اشاره کرد. پستگرس تنها یک سامپادرا نیست، بلکه یک سامانه مدیریت پای شی‌گرا-رابطه‌ای محسوب می‌شود (سامپاداراش).

معایب پستگرس

با استناد به ادعای پایگاه دیجیتال اوشن، در شرایطی که عملیات خواندن سنگین روی پستگرس اعمال شود، در رقابت با پایگاه‌داده‌هایی مانند مای‌اس‌کیوال، عمل‌کرد ضعیف‌تری از خود نشان می‌دهد.

زمان‌هایی که باید از پستگرس استفاده کنیم:

پستگرس ابزاری عالی برای مواقعی است که احتیاج به پایداری بالای اطلاعات داریم و تحت هیچ شرایطی نمی‌توان این موضوع را نادیده گرفت. در این مواقع، پستگرس نسبت به رقبا انتخاب بهتری است.

هم‌چنین در شرایطی که خدمت ما، طراحی بسیار پیچیده‌ای دارد، پستگرس نسبت به رقبا پیش است.

SQLight

نوع: سامپادرا (توکار)

اس‌کیوالایت یک سامپادرا با حجم بسیار کم است. اس‌کیوالایت به صورت یک خدمتِ جدا از برنامه اصلی که روی میزبان خود اجرا شود نیست، بلکه درواقع یک کتابخانه در زبان سی است که به عنوان جزئی از برنامه اصلی اجرا می‌شود. به همین دلیل، برخلاف دیگر انواع پایگاه داده‌ها، اس‌کیوالایت به مراتب سبک‌تر از همتایانش به حساب می‌آید.

اس‌کیوالایت تا حد خوبی، اکثر دستورهای SQL را پیاده‌سازی می‌کند، ولی تمامی ویژگی‌های SQL را ندارد. این پایگاه داده، تمامی اطلاعات مربوط را روی یک پرونده واحد نوشته و به همین دلیل، اجازه خواندن هم‌زمان از روی پایگاه داده‌ها را داریم، ولی اجازه نوشتن هم‌زمان داده‌ها روی پایگاه داده‌ها در دسترس نخواهد بود. با توجه به سبکی و این موضوع که فرآیندی جدید روی دستگاه اجرا نمی‌کند، اس‌کیوالایت به صورت گسترده در برنامه‌های سمت کاربر مورد استفاده قرار می‌گیرد.

برای مثال، مرورگرهای اینترنتی برای ذخیره تاریخچه بازدیدهای خود، از اس‌کیوالایت استفاده می‌کنند. هم‌چنین خود سیستم‌عامل‌ها به صورت گسترده از اس‌کیوالایت استفاده می‌کنند. با توجه به استفاده بسیار گسترده اس‌کیوالایت در برنامه‌های سمت کاربر، به نظر می‌رسد اس‌کیوالایت، پرکاربردترین پایگاه داده موجود باشد.

اس‌کیوالایت توسط دکتر ریچارد هیپ که در سال ۲۰۰۰ برای نیروی دریایی ایالات متحده کار می‌کرد، به وجود آمده است. در آن زمان، دکتر هیپ روی برنامه مربوط به موشک‌های هدایت شونده که از روی ناوهای دریایی شلیک می‌شدند، کار می‌کرده و تا پیش از اس‌کیوالایت، از پایگاه داده شرکت IBM با نام اینفورمیکس استفاده می‌کردند. هدف از ساخت این پایگاه داده این بود که برنامه‌ها، بدون نصب پایگاه داده، به صورت مستقل اجرا شوند.

در باب مزایا و معایب این پایگاه داده، در بالا تا حد خوبی اشاره شد. با توجه به تفاوت مفهومی استفاده از اس‌کیوالایت، طبیعتا این پایگاه داده، با پایگاه‌داده‌های معمول که در کارسازهای وب مورد استفاده قرار می‌گیرند، قابل قیاس نیست و به شدت ضعیف‌تر

عمـل می‌کنـد. در ضمن، با توجه بـه این موضوع که اس‌کیوالایـت اجـازه نوشـتن همزمـان روی خـود را نمی‌دهد، مطلقا گزینه جالبی برای سمت کارساز به حسـاب نمی‌آیـد. ولی با توجه به قـدرت بالا در عین سبکی، به شـدت در برنامه‌های سمت کارخواه مورد استفاده قرار گرفته، که با توجه به استفاده کم از منابع میزبـان در قیاس بـا دیگر پایگاه داده‌هـا، کاربردهای فراوانی در این حوزه دارد.

Neo4j

نوع: پایگاه داده گرافی

نئوفورجـی یـک پایگاه داده گرافی اسـت که توسـط جـاوا پیاده‌سـازی شـده اسـت. نئوفورجـی از زبـان توسـعه‌دهندگان خود آن چنین نامیده می‌شود: یک پایگاه داده توکارِ دیسک-محور و یک موتور مبادلاتی مانـدگارِ جـاوا، که داده‌ها را به جای این که به شـیوه معمول در جدول‌های رابطه‌مند ذخیره کند، در گراف ذخیره‌می‌کند.

نئوفورجی با دو پروانه ارائه می‌شود: نگارش جامعه‌محور آن با GPLv3 ارائه شـده و تعدادی از مولفه‌های اضافی آن با Affero GPL یا AGPL منتشـر می‌شوند. هم‌چنین نگارش تجاری آن نیز موجود است.

نگارش ۱.۰ نئوفورجی در ۲۰۱۰ انتشار یافت و نگارش ۲.۰ آن نیز در ۲۰۱۳ منتشر شد.

ایـن پایگاه داده که به عنوان محبوب‌ترین پایگاه داده گرافی شناخته می‌شود، در شـرکت Neo Technology inc واقع در سانفرانسیسکوی ایالات متحده و مالموی سوئد ساخته و توسعه داده شده است.

در نئوفورجـی همـه داده‌ها به صورت یـک یال، یک رأس یا یک مشخصه ذخیره می‌شود. هر رأس و یال می‌توانند به تعداد دل‌خواه مشـخصه داشـته باشـند. هـم یال‌هـا و هم رأس‌ها می‌توانند برچسـب بخورند. برچسب‌ها می‌توانند برای محدود کردن جست‌وجوها به کار روند.

از مزایای نئوفورجی می‌توان به این موارد اشاره کرد:

- یک ابزار خوب برای نشان دادن ارتباط میان داده‌ها است و با توجه به گراف محور بودن پایگاه داده، این کار بسیار ساده و قابل فهم انجام می‌شود.
- ایـن پایگاه داده بـرای انجام پیمایش‌های مختلف روی داده‌های مرتبط به هم، بسیار ساده و سریع عمل می‌کند
- از مدل داده‌های بسیار ساده‌ای بهره می‌برد.

در بـاب معایب این پایگاه داده هـم می‌توان به وجود برخی محدودیت‌ها در رأس‌های گراف اشاره کرد و این که نئوفورجی از سامانه sharding پشتیبانی نمی‌کند.

بـه صـورت خلاصـه از سـامانه sharding برای پخش کـردن بار پردازشـی سـامانه در فرایندهای حجیم و مقیاس‌پذیری افقی سامانه استفاده می‌شود.

Redis

نوع: ذخیره کلید-مقدار

ردیس یک پایگاه داده درون حافظه‌ای است. این پایگاه داده‌هـا از حافظه اصلی کارسـاز برای ذخیـره داده‌ها استفاده می‌کنند که موجب سرعت بیشتر پایگاه داده و استفاده کم‌تر از توان پردازشی کارساز می‌شود، هرچند که طبیعتا پایداری کم‌تری را نیز به همراه می‌آورد. با وجود فن‌آوری سـخت‌افزاری non-volatile random access memory حتـا در صورت قطع شـدن برق نیز، از داده‌ها محافظت می‌شود.

ردیـس از سـاختار کلید-مقـدار برای ذخیـره داده‌ها بهره برده و کلیدها در آن با پایداری اختیاری ذخیره می‌شـوند. به ایـن معنی که تضمین می‌شـود زمانی کـه یک تراکنش در پایگاه داده‌ها ثبت شـد، تا مدت مشـخص تعیین شـده، این تراکنش در سامانه باقی مانده و از بین نمی‌رود.

ردیس نخسـتین بار در ۲۰۰۹ منتشـر شـد و توسط VMware و Pivotal Software پشـتیبانی می‌شـد، ولـی از ۲۰۱۵ توسـط آزمایشـگاه‌های ردیس پشتیبانی و حمایت‌می‌شود.

از بُعد کارایی و سـرعت، در شـرایطی که احتیاجی به پایـداری داده‌ها نباشـد، ردیس فوق‌العاده سـریع‌تر از پایگاه‌داده‌های معمول که هر تغییری را روی دیسک ذخیره می‌کنند، عمل می‌کنـد. ولی در صورت فعال بودن پایداری داده‌ها، به این دلیل که داده‌ها به صورت مقطعی، روی دیسک سخت پشتیبان گیری می‌شوند، کارایی سامانه مقداری پایین می‌آید.

هم‌چنین به این دلیل که ردیس به صورت تک‌رشته‌ای و تک‌فرایندی عمل می‌کند، یک اجرا از ردیس، توانایی انجام وظایف موازی را ندارد.

اسـتفاده از حافظه اصلی در ردیس، با این که موجب سـرعت بیشتر می‌شـود، ولی با توجه به هزینه بالاتر حافظه اصلی نسبت به دیسک سخت، هزینه بیش‌تری روی دست کاربران خود می‌گذارد.

Cassandra

نوع: سامپاد توزیع‌شده

کاساندرا یک سامانه مدیریت پایگاه داده توزیع شده اسـت کـه برای پـردازش حجـم زیـادی از داده روی تعداد زیادی کارسـاز طراحی شـده است. کاساندرا با جاوا توسـعه داده شـده و دسترسـی بسـیار بالایی را بدون گلـوگاه نرم‌افزاری ارائه می‌کند به این معنا که در کاساندرا، هیچ بخشی پیدا نمی‌شود که درصورت به وجود آمدن مشـکلی در آن بخش، تمام سامانه از کار بیفتد.

آپاچی کاساندرا در ابتدا به منظور بهبود جست‌وجو در صندوق‌های پستی کاربران در فیس‌بوک ساخته شد و در ۲۰۰۸ بـه عنـوان یک پروژه آزاد روی گوگل کُد منتشر شد، و از ۲۰۰۹ تا کنون، توسط بنیاد آپاچی پشتیبانی‌می‌شود.

با توجه به کارایی بسیار بالای این پایگاه داده، بسیاری از شرکت‌های بزرگ فن‌آوری جزو کاربران آن هستند که در این میان، شرکت اپل با ۷۵۰۰۰ گره و ذخیره بیش از ۱۰ پتابایت داده، به عنوان بزرگ‌ترین شرکت استفاده کننده از این پایگاه داده، مطرح می‌شود.

هم‌چنیـن در فهرسـت دیگر اسـتفاده کننـدگان از کاساندرا، نام‌های معتبری چون نت‌فلیکس، ای‌بی، گو ددی، گیت‌هاب، اینسـتاگرم، سـرن و بیش از ۱۵۰۰ شرکت بزرگ دیگر، دیده می‌شوند.

کاساندرا کارایی بسیار بالایی ارائه می‌دهد. برای مثال در سال ۲۰۱۲، دانشگاه تورنتو طی تحقیقاتی که روی سامانه‌های NoSQL انجام داد، در حوزه مقیاس‌پذیری، کاسـاندرا را به عنـوان برنده بلامنـازع در میان دیگر پایـگاه داده‌ها اعلام کـرد. در طی آزمایش‌های آن‌ها، کاسـاندرا بیش‌ترین توان عملیاتی را برای بیش‌ترین تعـداد گـره در تمـام آزمایش‌ها ارائـه داد، گرچه این موفقیـت با هزینـه زمانی بالای خواندن و نوشـتن همراه بود.

کاسـاندرا، از مـدل داده ردیف‌هـای بخـش بنـدی شـده اسـتفاده کـرده و در آن، ردیف‌هـا در جدول‌ها دسـته‌بندی می‌شوند. کلید نخسـتین مولفه هر جـدول، کلید بخش آن اسـت و در داخل هر بخش،

ردیف‌ها توسط ستون‌های کلید باقی‌مانده، خوشه بندی شده‌اند.

HBase

نوع: سامپاد توزیع‌شده غیر رابطه‌ای

اچ‌بیس هم از دیگر پروژه‌های آپاچی است که به عنوان بخشی از پروژه هدوپ توسعه داده شده و روی بستر سامانه‌پرونده توزیع شده هدوپ (HDFS) اجرا می‌شود. اچ‌بیس یک پایگاه داده توزیع‌شده غیر رابطه‌ای است که پس از پروژه جدول بزرگ گوگل، تحت زبان جاوا ساخته و توسعه داده شده است. این پایگاه داده در واقع قابلیت‌های جدول بزرگ گوگل رابرای هدوپ به ارمغان آورده و یک روش تحمل‌پذیری خطا را برای داده‌هایی که توسط تعداد زیادی داده خالی و یا داده بی‌ارزش احاطه شده‌اند، ارائه می‌دهد.

اچ‌بیس نخستین بار در شرکت پاورست برای اهداف پردازش زبان طبیعی ساخته شد و اکنون توسط بنیاد آپاچی پشتیبانی می‌شود.

از ویژگی‌های اچ‌بیس می‌توان به فشرده‌سازی، انجام عملیات در حافظه کارساز و یا فیلتر بلوم به ازای هر ستون یاد کرد. این ویژگی‌ها براساس مقاله‌ای که درمورد جدول بزرگ از سمت گوگل انتشار یافت، پیاده سازی شده است.

جدول‌ها در اچ‌بیس به عنوان ورودی و خروجی برای کاهش نگاشت در هدوپ عمل می‌کنند.

اچ‌بیس به عنوان یک جایگزین سامانه‌های SQL بیان نمی‌شود، هرچند که اخیرا بهبودهای بسیار قابل توجهی روی کارایی آن انجام شده است و اکنون به عنوان پایگاه داده بسیاری از پایگاه‌های وب داده‌محور، از جمله خدمت پیام‌رسانی فیس‌بوک عمل می‌کند.

Microsoft SQL Server

نوع: سامپادرا

اس‌کیوال سرور مایکروسافت یک سامپادرا است که با سی/سی‌پلاس‌پلاس نوشته شده است. این پایگاه داده، یک نرم‌افزار آزاد نبوده و متعلق به مایکروسافت است و برخلاف دیگر پایگاه داده‌ها، روی تمامی سیستم‌عامل‌ها اجرا نشده و فقط روی بن‌سازه‌های ویندوزی اجرا می‌شود.

در سال ۱۹۸۸، مایکروسافت به شرکت‌های استون تِیپ و سی‌بیس پیوست تا نوع دیگری از پایگاه داده سی‌بیس اس‌کیوال سرور را برای IBM OS/2 بسازد. این پایگاه داده، نخستین نگارش اس‌کیوال سرور بود و موجب ورود مایکروسافت به بازار فروش پایگاه داده‌ها و آغاز رقابت با اوراکل و IBM و بعدها سی‌بیس شد.

اس‌کیوال سرور مایکروسافت در ۷ نگارش اصلی به این نام‌ها منتشر شده است: مرکز داده، شرکتی، استاندارد، وب ،کسب‌وکار، کارگروهی و نگارش اکسپرس. هم‌چنین نگارش‌های خاص دیگری نیز وجود دارد. برای مثال می‌توان به نگارش‌های توسعه‌دهنده، یا نگارش آژر که به عنوان یک سامانه ابری از سوی مایکروسافت استفاده می‌شود، اشاره کرد.

برای اس‌کیوال سرور، یک لایه رابط خارجی طراحی شده است که موجب می‌شود هر عملیاتی را که بخواهیم با آن انجام دهیم، با استفاده از قالب از پیش تعریف شده مایکروسافت به نام TDS استفاده کند. TDS یک لایه نرم‌افزاری است که برای تبادل اطلاعات بین پایگاه داده‌ها و کاربر به‌کار می‌رود.

Hive

نوع: پایگاه داده زیرساخت

هایو یک پایگاه داده زیرساخت است که برای ارائه خلاصه‌ای از داده‌ها و تحلیل آن‌ها روی هدوپ اجرا می‌شود. هایو نخستین بار توسط فیس‌بوک ایجاد شد، ولی اکنون توسط دیگر شرکت‌ها مانند نت‌فلیکس و یا آمازون توسعه می‌یابد.

هایو از تحلیل تعداد بسیار زیادی داده که روی HDFS ذخیره شده‌اند پشتیبانی می‌کند. البته اکنون از سامانه‌پرونده Amazon s3 نیز پشتیبانی می‌کند.

هایو به صورت پیش‌گزیده فراداده‌ها را به صورت توکار در آپاچی دربی یا هر پایگاه داده مای‌اس‌کیوال مانندی ذخیره می‌کند.

هایو برای سرعت بیش‌تر، تمامی نوع داده‌ها، حتا داده‌های نگاشت بیتی را شاخص‌گذاری می‌کند. هم‌چنین هایو اطلاعات مربوط به خود را روی پرونده یا حتا روی دیگر پایگاه‌داده‌ها مانند اچ‌بیس ذخیره می‌کند.

هایو برای انجام عملیات روی داده‌ها، از الگوریتم‌های BWT و اسنپی استفاده می‌کند.

زبان پرس‌وجوهای هایو بسیار شبیه SQL بوده و به سادگی قابل تبدیل به پرس‌وجوهای اسپارک و تز است.

solr

نوع: واسط برنامه‌نویسی جست‌وجو و نمایه‌گذاری

سولر یک بستر آزاد برای جست‌وجو در پایگاه داده‌ها است که با جاوا نوشته شده و منشأ اصلی آن، پروژه لوسین آپاچی بوده است. ویژگی‌های اصلی آن، جست‌وجو بین متن‌ها و نمایه‌گذاری آنی محتوا است و حتا بین پرونده‌های متنی موجود در پایگاه داده‌ها مثل pdf و پرونده‌های اداری نیز جست‌وجو می‌کند. سولر یک سامانه جست‌وجوی توزیع‌شده است و برای مقیاس‌پذیری و تحمل خطا طراحی شده است. لازم به ذکر است که سولر معروف‌ترین موتور جست‌وجوی تجاری نیز محسوب می‌شود.

Elasticsearch

نوع: جست‌وجو و شاخص‌گذاری

الستیک‌سرچ یک جست‌وجوگر کارساز است که با استفاده از جاوا و بر مبنای پروژه لوسین ساخته شده و یک موتور جست‌وجوی توزیع‌شده با ظرفیت چندگانه است. الستیک‌سرچ، پس از سولر، دومین موتورجست‌وجوی سمت کارساز تجاری معروف به حساب می‌آید.

الستیک‌سرچ نیز قابلیت جست‌وجو میان تمامی پرونده‌ها و اسناد اداری، pdf و دیگر اسناد متنی را دارا بوده ویک موتور جست‌وجوی توزیع شده است. به این معنا که تعدادی نمایه به shardهای مختلف تقسیم می‌شوند و هر shard می‌تواند صفر و یا بیشتر از خودش رونوشت بگیرد. در کارساز، هر گره، یک یا بیش از یک shard را میزبانی می‌کند و به عنوان هماهنگ کننده، هر عملیات را به shard مورد نظر هدایت می‌کند.

درمیان کاربران الستیک‌سرچ نام‌های معتبری چون ویکی‌مدیا، گیت‌هاب، موزیلا، کیورا، نت‌فلیکس، سرن و... دیده می‌شود.■

BASE در مقابل ACID

در علم شیمی pH یک کمیت برای مشخص کردن نسبت اسیدی یا بازی بودن یک محلول در آب است. تغییرات pH در بازه ۰ تا ۱۴ قرار دارد و محلول بشدت اسیدی مانند اسید باطری عدد ۰ و محلول بشدت بازی مثل lie عدد ۱۴ را کسب می‌کند. این در حالی است که آب در دمای ۲۵ درجه سلسیوس دارای pH ۷ است و این حالت را خنثی می‌نامند.

مهندسین داده با هوشمندی این استعاره را از شیمی‌دان‌ها وام گرفتند و آن را به یک اختصار تبدیل کرده‌اند. البته شاید معنی واقعی خود را نداشته باشد، اما زمانی که بحث درباره قابلیت اطمینان پردازش کردن تراکنش است همچنان نشان‌دهنده رویدادهای مربوط به یک سیستم پایگاه داده است. برای گسترش دادن بیشتر این استعاره مفید اجازه بدهید اول نگاهی به ایده pH بکنیم.

اولین بار «سورنسن» در حالی که در آزمایشگاه کارلسبرگ در ۱۹۹۰ در حال کار بود، این مفهوم را توسعه داد. بنابراین ایده مربوط به زمان حاظر نیست. در این‌جا حرف «H» به اتم‌های هیدروژن اشاره دارد و در مورد حرف «p» بحث و اختلاف فراوانی وجود دارد. برخی آن را از واژه فرانسوی «puissance»، برخی آن را از واژه آلمانی «potenz» و برخی دیگر آن را از واژه لاتین «pondus» یا «potentia» می‌دانند. اما هرچه باشد همه این‌ها به معنی پتانسیل یا توان (power) است. بنابراین pH به معنای قدرت، پتانسیل یا فعالیت اتم‌های هیدروژن در محلول است. محلول‌های اسیدی فعالیت بیشتری دارند و محلول‌های قلیایی کمتر فعالند.

در پردازش کردن تراکنش پایگاه‌داده‌ها، داده‌ها اتم هیدروژن هستند که در سیستم جریان دارند و حرف «p» ویژگی‌هایی است که پردازش شدن این تراکنش‌ها را با داشتن قابلیت اطمینان تضمین می‌کند.

ACID چیست؟

اگر از هر کسی که در حوزه داده‌ها حرفه‌ای است این سوال را کنید، احتمالا به خوبی می‌تواند ACID (Atomicity, Consistency, Isolation, و Durability) را شرح دهد. مفهموم ACID برای سالیان وجود داشته است و تا همین اواخر نیز یک معیار اصلی بود که همه پایگاه‌های داده در تلاش برای کسب کردن آن بودند. اگر پایگاه‌داده‌ای ACID را نمی‌داشت، قابلیت اطمینان آن زیر سوال می‌رفت.

جیم گری اولین بار در سال ۱۹۷۰ این ایده را به تصویر کشید و پس از آن مقاله بدیع «مفهمو تراکنش: خواص و محدودیت‌ها» را در ژوئن ۱۹۸۱ منتشر کرد. در این مقاله تنها اصطلاحات تمامیت (یا همه یا هیچ)، یکپارچگی و پایائی مورد بحث قرار گرفته بودند و پس از آن بود که اصطلاح انزوا نیز اضافه شد.

در آن مقاله بر روی سطح تراکنش تمرکز شده بود و تراکنش را به عنوان قرارداد یا هر تعداد از «تراکنش‌های حالت سیستم» که مجبور به داشتن CAD به عنوان خواص ذاتی یا «محدودیت‌های قابلیت اطمینان سیستم» بودند، مشخص کرده بود.

«بروس لیندی» و همکارانش مقاله‌ی «یادداشت‌هایی بر پایگاه داده‌های توزیعی» را در ۱۹۷۹ منتشر کردند که بر روی کار «گری» ایجاد شده بود. او اصول را برای دستیابی به قابلیت اطمینان و استانداردهای اولیه تکرار پایگاه داده ایجاد کرد. در سال ۱۹۸۳ «آندریاس روتر» و «تئو هاردر» مقاله «مفاهیم بازیابی پایگاه‌داده‌های تراکنش‌گرا» را منتشر کرد و به طور رسمی اصطلاح ACID را به کار برد. این اصطلاح دو دهه بعد به صورت زیر معنی شد:

تمامیت: هیچ وظیفه (یا همه وظایف) درون یک تراکنش اجرا نمی‌شوند مگر این‌که همه آن‌ها انجام شوند. این اصل «همه یا هیچکس» است، اگر یکی از اجزای تراکنش با شکست روبرو شد همه تراکنش‌ها با شکست روبرو می‌شود.

قابلیت اطمینان: تراکنش باید تمام پروتکل‌ها یا قوانین تعریف شده توسط سیستم در تمام زمان‌ها را تامین کند. تراکنش نباید آن پروتکل‌ها را نقض کند و پایگاه‌داده‌ها باید در یک حالت پایدار در آغاز و پایان تراکنش بمانید. هرگز هیچ گونه تراکنش نیمه انجام شده‌ای وجود ندارد

انزوا: هیچ تراکنشی به تراکنش دیگر که در یک حالت میانی و ناتمام است دسترسی ندارد. بنابراین هر تراکنش به خودی خود مستقل است. این موضوع برای کارایی و انزوای تراکنش‌ها در یک پایگاه‌داده‌ها نیاز است.

پایائی: زمانی که یک تراکنش کامل شد، در حالت انجام شده باقی می‌ماند و امکان بازگشت آن وجود ندارد. این تراکنش از شکست سیستم، قطع شدن برق و انواع دیگر از کارافتادگی سیستم مصون خواهد بود.

جنبه‌های بسیار دیگری برای این تعاریف وجود دارد و تامین نیازمندی‌های واقعی ACID مختص به هر پایگاه‌داده‌ها خاص است. اما در کل در دنیای RDMS همه چیز ACID است و بدون ACID قابلیت پایداری زیر سوال می‌رود.

عدد pH مربوط به ACID پایین است، تقریبا چیزی شبیه به اسید باطری (۰) و یا شاید سرکه (۲)، در این حالت داده و محدودیت‌های آن بسیار فعال هستند. بنابراین در هر میکروثانیه از یک پایگاه‌داده‌ها که از ACID به عنوان سیستم محدودیت استفاده می‌کند، همه داده‌ها (اتم‌های هیدروژن) تحت بررسی ثابت هستند تا از این برآورده‌شدن محدودیت‌هااطمینان حاصل کنند. چنین محدودیت‌هایی برای سالیان در دنیای رابطه‌ای،نرمال، طرح واره محور، مقیاس پذیر افقی و کوچک پیش از شبکه‌های اجتماعی به خوبی کار می‌کرد.

این بدیهیات گذشته دیگر مطرح نیستند و اکنون داده بدون ساختار، بزرگ‌داده، ساختارهای داده غیر رابطه‌ای، سیستم‌های محاسباتی توزیع شده و قابلیت اطمینان نهان اکنون متداول شده‌اند. این نیازمندی‌های جدید به کلمات اختصاری جدید و pH جدید نیاز دارد.

قضیه CAP:

در سال ۲۰۰۰ «اریک برووِر» (Eric Brewer) سخنرانی خود را در همایش ACM که در مورد مفاهیم محاسبات توزیع شده بود ارائه داد و آن‌جا بود که قضیه CAP متداول شد.

اکثر مردم در جهان خارج هیچ‌گاه حتی اسم این قضیه را نشنیده‌اند و اهمیتی هم برایشان ندارد. آن‌ها فقط می‌خواهند رایانه‌شان کار کند، اینترنت کار کند، شبکه‌های اجتماعی کار کنند و البته همه این‌ها با در دسترس بودن قابلیت اطمینان فایل‌هایشان.

قضیه CAP، که با نام قضیه برووِر هم شناخته می‌شود، بعدها تجدید نظر شد و از طریق کار «شیت گیلبرت» و «نانسی لینچ» از MIT در سال ۲۰۰۲ تغییر کرد.

اصل اساسی قضیه می‌گوید که سه نیازمندی اساسی در سیستم وجود دارد که این اصول برای یک طراحی، پیاده‌سازی و استقرار موفق برنامه‌ها در سیستم‌های محاسباتی توزیع شده ضروری هستند. این نیازمندی‌ها قابلیت اطمینان، دسترس‌پذیری و تحمل قسمت‌بندی است، که به اختصار CAP نامیده می‌شود.

قابلیت اطمینان به این مساله اشاره می‌کند که آیا یک سیستم کاملا عمل می‌کند یا نه؟ آیا قابلیت اعتماد سیستم از قوانین مشخص شده در برنامه‌نویسی‌اش با توجه به این مقررات تعریف شده پیروی می‌کند؟ آیا همه گره‌ها در یک خوشه، همه داده‌های مفروض برایشان را می‌بینند؟ این همان ایده معرفی شده در ACID است.

دسترس‌پذیری به معنای همان چیزی است که انتظارش می‌رود. آیا سیستم یا سرویس داده شده زمانی که درخواستی به آن ارسال می‌شود، حاضر است؟ آیا هر درخواست پاسخی خارج از شکست یا موفقیت را دریافت می‌کند؟

تحمل قسمت‌بندی، نشان‌دهنده این حقیقت است که یک سیستم داده شده، در شرایط از دست رفتن اطلاعات یا خرابی سیستم به عمل کردن ادامه می‌دهد. شکست یک گره نباید باعث سقوط کل سیستم شود.

این‌ها تنها تعاریف ساده‌ای از سه جنبه قضیه CAP بودند. مقالات فراوانی در این باره وجود دارد که بسیاری از تفاسیر، تحلیل‌ها و مشکلات پیچیده موجود در کاربردهای دنیای واقعی این قضیه را بحث کرده‌اند. دلیل اصلی معرفی قضیه این نکته است که در اکثر نمونه‌ها و سیستم‌های توزیع شده تنها می‌توانند دو ویژگی از این ویژگی‌ها را تضمین کنند. در این‌جا چشم‌پوشی می‌تواند نتایج فاجعه باری داشته باشد که شامل احتمال شکست خوردن تمام سه عامل به صورت همزمان و جداگانه است.

محدودیت‌های قضیه CAP با قابلیت اطمینان پایگاه‌داده‌ها برای سیستم‌های بزرگ مقیاس، توزیع‌شده، غیررابطه‌ای همچنان پابرجا است. بیشتر اوقات آن‌ها به دسترس‌پذیری و تحمل قسمت‌بندی نیاز دارند. بنابراین قابلیت اطمینان فدا می‌شود و ACID سقوط می‌کند. یک عبارت مناسب این‌جا این است که «برای تپه‌ها بران» (هرچه می‌توانی برو، سریع برو)

BASE خودش را معرفی می‌کند

خوشبختانه در دنیای سیستم‌های محاسباتی توزیع شده، مهندسین باهوش وجود دارند. چگونه سیستم‌های گسترده جهانی مانند بیگ‌تیبل گوگل (Google's BigTable)، آمازون دینامو و کاساندرای فیسبوک با از دست دادن قابلیت اطمینان سر و کله می‌زنند و قابلیت اطمینان سیستم را نیز حفظ می‌کنند؟

پاسخ این سوال در حالی که قطعا ساده نیست، اما جواب BASE (Basically Available, Soft state, Eventual consistency) است.

در یک سیستم در حالی که BASE نیازمندی اصلی برای قابلیت اطمینان است، فعالیت یا پتانسل (p) از تغییرات داده (H) اساسا آرام کردن آن است. در مقیاس گذاری pH، یک BASE نزدیک‌تر به آب صابونی است (۱۲) یا شاید نمک دریاچه بزرگ (۱۰).

ادعای چنین اظهار نظری این نیست که میلیون‌ها تراکنش به سرعت رخ نمی‌دهند، بلکه رخ می‌دهند و تنها محدودیت‌های اعمال شده بر آن‌ها تغییر کرده است. آن محدودیت‌ها در زمان‌های متفاوت با قوانین متفاوت رخ می‌دهند.

اساسا در دسترس: این محدودیت بیان می‌کند که این سیستم آماده بودن داده‌ها را با توجه به قضیه CAP تضمین می‌کند و یک پاسخ به هر درخواستی وجود دارد. اما این پاسخ می‌تواند «شکست» برای داده درخواست شده باشد، و یا ممکن است این داده‌ها در یک وضعیت نامناسب یا در حال تغییر باشد. این مساله مشابه اتفاقاتی است که برای پاس شدن یک چک ممکن است اتفاق بیفتد.

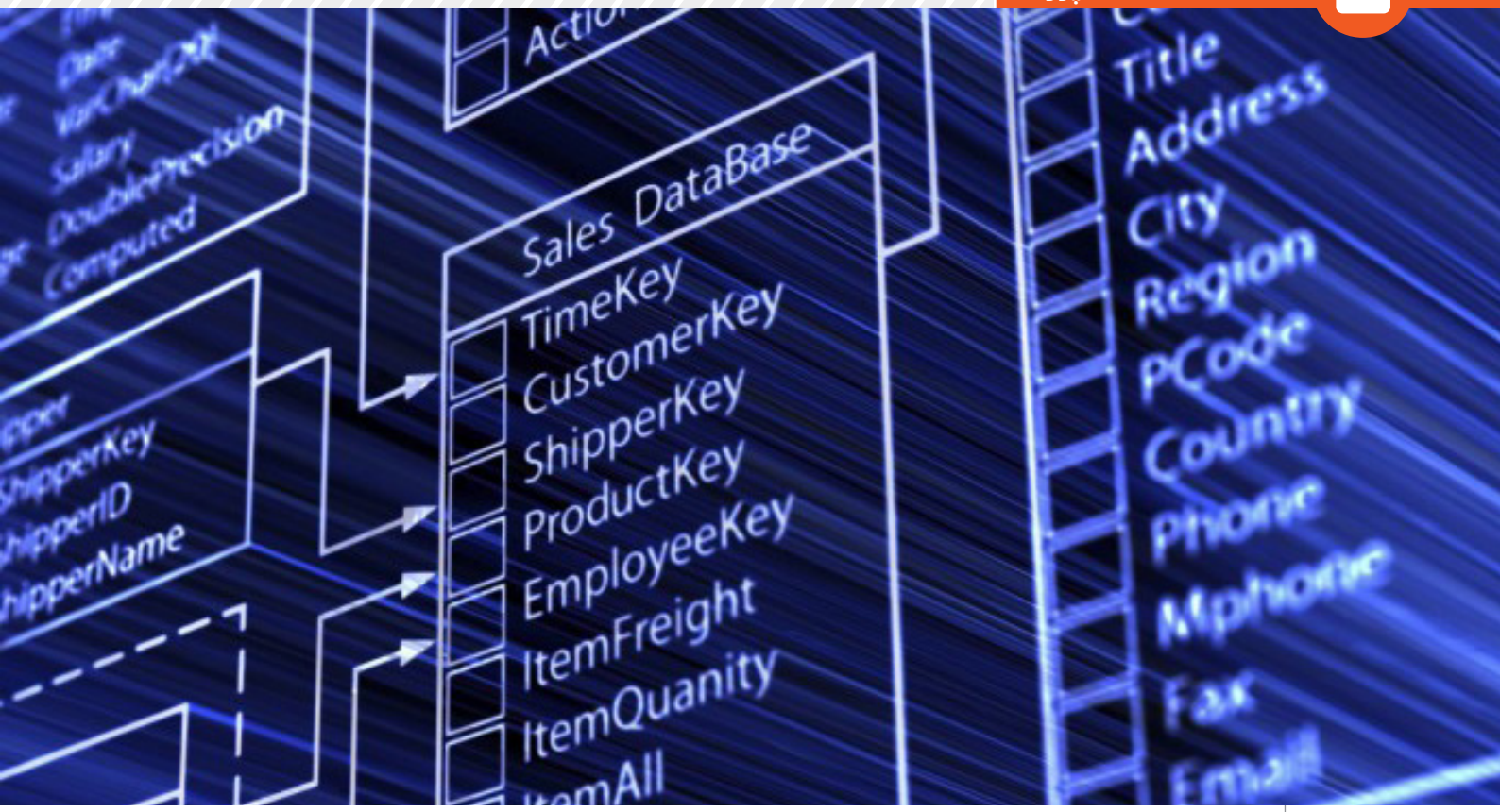
حالت نرم: حالت سیستم می‌تواند در طول زمان تغییر کند. حتی در طول زمانی که هیچ ورودی وجود ندارد ممکن است تغییرات برای رسیدن به «قابلیت اطمینان نهایی» انجام شود. بنابراین این حالت از سیستم همیشه «نرم» است.

قابلیت اطمینان نهایی: سیستم در نهایت قابلیت اطمینان خواهد شد، پس از آن که دریافت ورودی به پایان برسد. دیر یا زود داده به همه‌جا گسترش پیدا می‌کند، اما سیستم به دریافت ورودی ادامه می‌دهد و قابلیت اطمینان هر تراکنش را پیش از آغاز تراکنش بعدی بررسی می‌کند. جزئیات کامل این موضوع در مقاله «قابلیت اطمینان نهایی- بازبینی شده» آمده است.

نتیجه‌گیری: حرکت به جلو

pH جدید پردازش کردن تراکنش پایگاه‌داده‌ها، امکان مقیاس‌گذاری عمومی را با سطوح مقرون به صرفه کارآمد می‌کند. بررسی قابلیت اطمینان هر تراکنش منفرد در هر لحظه از هر تغییر هزینه عظیمی را به یک سیستم که دارای ترلیون‌ها تراکنش است اضافه می‌کند. نیازمندی‌های محاسباتی حتی بیشتر نجومی هستند

«قابلیت اطمینان نهایی» این امکان را به سازمان‌هایی چون یاهو!، گوگل، توییتر، آمازون و صدها یا میلیون‌ها سازمان دیگر می‌دهد تا با کاربرانشان در سراسر جهان تعامل داشته باشند، در حالی که هزینه‌های خود را پایین نگه می‌دارند، سیستم‌هایشان کار می‌کند و کاربرانشان خوشحال هستند. علاوه بر این‌ها خواهان داشتن قابلیت اطمینان کامل همیشگی هستند. اما همان طور که «دن پری‌چت» در مقاله «BASE یک جایگزین برای ACID» بیان کرده است، باید توازنی وجود داشته باشد و قابلیت اطمینان نهایی برای توسعه موثر سیستم‌ها با وجود افزایش نمایی داده حاصل از، شبکه‌های اجتماعی، محاسبات ابری و سایر پروژه‌های بزرگ داده وجود داشته باشند.■



فهرست گذاری راه‌حلی برای دسترسی سریع به اطلاعات به جای پویش سطر به سطر پایگاه داده؛

فهرست‌گذاری در پایگاه‌داده چگونه کار می‌کند؟

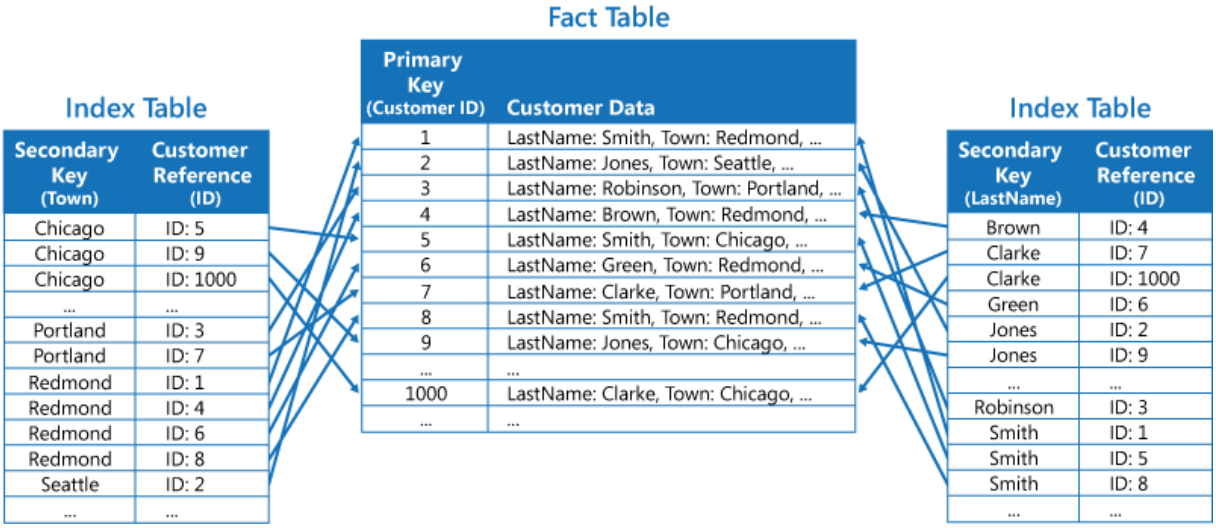
فهرست‌گذاری روشی در پایگاه‌های داده است که برای دسترسی به اطلاعات جداول و رکوردهای ذخیره شده در یک پایگاه‌داده‌ها به کار می‌آید. این روش، روشی ساده است که در زندگی روزمره ما نیز مورد استفاده است. در این روش ما جدولی از تمامی داده‌ها و شاخص آنان را می‌سازیم و برای دسترسی به اطلاعات ابتدا به آن مراجعه کرده و با پیدا کردن شاخص به آن شاخص مراجعه کرده و اطلاعت را می‌خوانیم. در ویکی‌پدیا فارسی تعریفی از این مفهوم به شکل کلی بیان شده‌است که در زیر بیان می‌شود.

«فهرست‌گذاری (Indexing) در پایگاه‌داده‌ها روشی برای ارایه دسترسی سریع به مقادیر یک یا چند ستون است. این بهبود در بازیابی، در ازای هزینه بیشتر در هنگام تغییر/درج/حذف رکوردها است. به بیانی دیگر، استفاده از فهرست، فرایند بازیابی (Retrieval) را سرعت می‌بخشد، ولی در هنگام حذف و اضافه و یا تغییر مقادیر، هم داده‌ها باید تغییر کند و هم فهرست باید به‌روز شود، و این به معنای هزینه بیشتر است.

به عنوان مثال، برای به‌دست آوردن رکورد حساب با داشتن شماره حساب، به جای آن که در بین تمامی حساب‌ها جستجو شود، سامانه پایگاه‌داده‌ها می‌تواند ابتدا به فهرست برای یافتن شماره بلوک دیسکی که رکورد مرتبط در آن است مراجعه کند و سپس تنها به آن بلوک دیسک مراجعه کرده و رکورد را استخراج کند.»

بیایید موضوع را با بحثی آغاز کنیم که اصولا چرا باید از فهرست‌گذاری در پایگاه‌های داده استفاده کنیم. برای فهم بیشتر این موضوع از مثالی ساده شروع می‌کنیم. فرض کنید که ما دارای جدولی در یک پایگاه‌داده‌ها فرضی و با نام کارمند «Employee» ساخته‌ایم؛ این جدول دارای سه ستون است که هر یک از ستون‌ها مشخص کننده ویژگی‌های خاصی از یک کارمند است که به صورت ساده‌ای نام، سن و آدرس کارمندان در این فیلدها ذخیره می‌شود. به طور مشخص هر فیلد دارای نام‌های «Employee_Name»، «Employee_Age» و «Employee_Address» در بانک اطلاعاتی هستند. حال اگر بخواهیم با استفاده از یک پرس‌وجو یا کوئری (Query) ساده به اطلاعات داخل آن جدول دست پیدا کنیم که نام آنان عیسی «Jesus» است باید پرس‌وجوی اس‌کیوال زیر را اجرا کنید.

```
SELECT * FROM Employee WHERE Employee_Name = 'Jesus'
```



به‌طور دقیق‌تر زمانی که ما پرس‌وجوی فوق را برای جست‌وجو در جدول مذکور برای یافتن فردی با نام عیسی جستجو می‌کردیم، در پشت صحنه دقیقا چه اتفاقی در حال رخ دادن بوده‌است؟ در این زمان سامانه مدیریت پایگاه‌داده‌ها مجبور است که تمامی سطور و افراد موجود در جدول را یکی به یک بگردد تا این که سطری که فیلد نام کارمند در آن عیسی بود را بیابد. همچنین به این خاطر که ما تمامی کارمندانی را که نامشان عیسی است را می‌خواهیم؛ سامانه مدیریت پایگاه‌داده‌ها باید تمامی اطلاعات را یک به یک بگردد. ممکن است در جدول فوق چندین سطر با مشخصه نام کارمند عیسی وجود داشته باشد که نمی‌توان فقط به یک‌بار جستجو و یافتن یک فرد اکتفا کرد. در چنین مواقعی که ما نیازمند جستجو در یک جدول به دنبال فرد خاص هستیم باید در دیگر سطرها نیز به دنبال فرد مورد نظر بگردیم. در این پرس‌وجو سامانه مدیریت پایگاه‌داده‌ها از سطر اول تا سطر آخر در جدول را به دنبال کارمندان که با نام عیسی هستند، گشته و تمامی آنان را خواهد یافت. بنابراین برای جستجوی چند فرد با نام عیسی، تمامی جدول و اطلاعات ذخیره شده یک به یک توسط مدیر پایگاه‌داده‌ها پویش شده‌اند. در چنین مواقعی است که می‌گوییم یک پویش کامل از جدول فوق صورت گرفته‌است.

فهرست شاخص‌ها چه تاثیر بر کارایی پایگاه‌داده‌ها دارند؟

ممکن است تصور کنید که چنین روش جستجویی در جداول پایگاه‌های اطلاعاتی بسیار غیر بهینه هستند. طبیعتا این روش به این شکل کارایی پایینی داشته و همانند جستجو در فهرست قبول‌شدگان کنکور در روزنامه‌ها به صورت پویش از بالا و خط به خط است که جست‌وجو و پیدا کردن اطلاعات در آن ممکن بهینه نیست. این درست همان جایی است که مبحث فهرست‌گذاری وارد عمل می‌شود و فهرست‌ها مطرح می‌شوند. همان طور که از نام این مطلب مشخص است، فهرست‌ها در بانک‌اطلاعاتی راه‌حل مشکل فوق است. این روش خواهد توانست جست‌وجو در پایگاه‌داده‌ها را بسیار سریع‌تر کند. نکته اساسی و کلیدی در فهرست‌گذاری همان قطع جست‌وجوی خط به خط و پویش تمام جداول برای یافتن گزینه دلخواه است که با فهرست کردن و فهرست‌گذاری اطلاعات باعث دستیابی سریع به اطلاعات مورد جست‌وجو می‌شود.

فهرست و شاخص چیست؟

برای درک بیشتر این موضوع که فهرست در پایگاه‌داده‌ها دقیقا چیست، بیایید یک دفترچه تلفن را در نظر بگیریم. برای یافتن اطلاعات و افرادی که در آن قرار دارند می توان آن افراد را دسته‌بندی کرد. معمولا در یک دفترچه تلفن ما آنان را در شاخه‌هایی با اسم اول فامیل آنان مشخص می‌کنیم که از الف تا ی دسته‌بندی شده است. حال در فهرستی از تمامی دسته‌بندی‌های انجام شده در اول آن مشخص شده‌است که افرادی که حرف اول نام خانوادگی آنان «ت» است در صفحات ۶ تا ۸ هستند. این فهرست برای دستیابی به اطلاعات ایجاد شده است و با دسته‌بندی آنان و معرفی مکان هر دسته، یافتن اطلاعات را آسان‌تر کرده‌ایم.

فهرست‌گذاری و شاخص‌ها در پایگاه‌های اطلاعاتی نیز به این شکل عمل می‌کنند. به این شکل که جدولی از شاخص‌ها و دسته‌بندی‌ها ایجاد می‌شود که شامل محل هر شاخص است؛ سپس اگر در پایگاه‌داده‌ها به دنبال نام شخصی بگردیم همانند مثال مذکور، ابتدا به جدول فهرست رفته و با یافتن دسته مورد نظر، تنها به جستجوی آن دسته خواهد پرداخت و دیگر نیازی به پویش تمامی اطلاعات در پایگاه‌داده‌ها وجود نخواهد داشت. البته مورد مذکور فقط یک روش از فهرست‌گذاری است و روش‌های دیگری نیز برای این موضوع وجود دارد اما هدف و ساختار کلی آنان به یک شکل است. با وجود این که استفاده از فهرستی از شاخص‌ها باعث بهینه شدن پایگاه‌داده‌ها و کارایی بیشتر در زمان جستجو خواهد بود. اما در برخی موارد بهتر است از این روش استفاده نشود. به طور کلی در استفاده از فهرست‌گذاری باید موارد زیر را در نظر داشت.

هنگامی که با جداول نسبتا کوچک سر و کار دارید شاخص‌ها کارایی زیادی ندارند.

به صورت کلی شاخص‌ها عملکرد را زمانی بهبود می بخشند که بر روی ستون‌هایی تعریف شوند که در الحاق «Join» جداول استفاده شده است.

چه نوع از ساختارهای داده‌ها را می‌توان به عنوان فهرست در نظر گرفت؟

در فهرست‌ها اطلاعات معمولا به شکل درخت بی (B-tree) نگاهداری می‌شود. بر اساس تعریفی که در ویکی‌پدیا فارسی از درخت بی ذکر شده است، در علوم کامپیوتر، یک درخت بی یا بی‌تری داده‌ساختاری درختی است که داده‌ها را به صورت مرتب‌شده نگه می‌دارد و جستجو، درج و حذف را در زمان مصرفی لگاریتمی میسر می‌سازد. بر خلاف درخت‌های جستجوی دودویی متوازن (Balanced binary search tree)، این داده‌ساختار برای سیستم‌هایی که بلاک‌های عظیم اطلاعات را خوانده و می‌نویسند بهینه‌سازی شده است. این ساختارداده معمولا در پایگاه‌های داده‌ها و سیستم پرونده استفاده می‌شود. یکی از دلایلی که باعث شده‌است که استفاده از درخت بی در جداول فهرست‌ها است به دلیل کارایی بالای آنان و تاثیری است که بر زمان جست‌وجو، حذف و اضافه از آن است که تمامی این اعمال در زمانی لگاریتمی انجام می شوند. از دلایل مهم دیگری که درخت بی را به یک روش محبوب تبدیل کرده است می‌توان به امکان مرتب بودن داده‌ها در آن اشاره داشت. به طور کلی سامانه مدیریت پایگاه‌داده‌های رابطه‌ای، تعیین کننده ساختاری است که برای فهرست‌ها استفاده می‌شوند اما در برخی شرایط برای برخی برخی از سامانه‌های مدیریت پایگاه‌داده‌های رابطه‌ای می‌توانید مشخص کنید که چه ساختمان‌داده‌هایی را می‌خواهید در زمان ایجاد فهرست در پایگاه اطلاعاتی خود ایجاد کنید.

فهرست جدول هش چگونه کار می‌کند؟

در علوم رایانه، جدول درهم‌سازی یا جدول «هش» نوعی ساختمان داده‌ها است که مقدارهایی که باید ذخیره شوند را به وسیله تابع هش (درهم‌سازی) با کلیدهای ویژه‌ای مرتبط می‌سازد. عملیات اولیه‌ای که این جدول تسهیل می‌کند عمل مراجعه است. به این معنی که کاربر می‌تواند با سرعتی کارآمد داده مورد نظرش را در آن بیابد. در جدول‌های هش همچنین افزودن داده‌های جدید در زمان کم امکان‌پذیر است. زمان لازم برای جستجو و افزودن وابسته به نوع جدول و میزان داده‌ها هستند. این زمان می‌تواند با انتخاب جدول مناسب به مرتبه زمانی O برسد.

جداول هش نیز نوع دیگری از ساختمان‌داده‌هاست که ممکن است برای فهرست‌گذاری برخی فهرست‌ها «Indexes» به کار رفته باشد. این فهرست‌ها معمولا با نام فهرست‌های هش شناخته می‌شوند. یکی از دلایل که باعث استفاده از هش در فهرست شده‌است، به این دلیل است که در زمان‌هایی تنها به جستجو در مقادیر نیاز داریم. استفاده از هش بسیار بهینه‌تر از دیگر روش‌ها است. بنابراین اجرای کوئری یا پرس‌وجوی دلخواه برای یافتن رشته‌ای مساوی رشته خاص و مورد نظر ما همانند مثال مذکور در اوایل مطلب بسیار سریع‌تر خواهد بود. برای نمونه اجرای کوئری ابتدای مقاله با استفاده از فهرست هش بر روی نام کارمند از جدول هش منفعت برده و با سرعت بهتری به نتیجه دلخواه خواهد رسید. به خاطر این موضوع، جداول هش به طور گسترده در نرم‌افزارها، به خصوص در آرایه‌های شرکت پذیر، ساختار فهرست پایگاه‌داده، حافظه‌های نهان و مجموعه‌ها استفاده می‌شوند.

در جداول هش معمولا به این شکل است که اطلاعات و مقادیر هر فیلد و رکود را در یک آرایه که به سادگی جدول هش نام دارد می‌گذارد که دارای مشخصه‌های کلید، مقدار و طول است. جدول هش اندیس یک کلید را محاسبه می‌کند و از آن اندیس برای جای دادن آن کلید در جدول استفاده می‌کند. به طور کلی تصور کنید که می‌خواهید اطلاعات چندین‌هزار نفر را در یک پایگاه‌داده‌ها ذخیره کنید؛ راه‌حل ابتدایی و ساده این است که تمامی آنان را در یک جدول ساده ریخته و ذخیره کنیم. بعد این همان طور که ذکر شد در صورتی که بخواهیم به دنبال فردی بگردیم باید n بار که n تعداد افراد داخل پایگاه‌داده‌ها است را یک به یک بگردیم. در این صورت مرتبه زمانی n خواهد بود. راه‌حل جدول هش در این زمان با ایجاد کلیدی که از آدرس مقادیر ذخیره شده را در پایگاه داده به صورت آدرس حافظه‌ای محل ذخیره نام‌ها به بدست آورده است و ذخیره آن به عنوان کلید در جدول هش کلیدی را به هر یک از آنان نسبت می‌دهد سپس نام آنان را نیز در کنار هریک از کلیدها متناسب با نام قرار می‌دهد. مثلا کلید و نام متناظری که برای فرد مذکور با نام عیسی ساخته شده‌است به این شکل در جدول هش قرار می‌گیرد.

```
“Jesus => 0x28939
```

برای جستوجو و یافتن فردی با نام عیسی در جدول با استفاده از کوئری نوشته شده در اوایل مطلب، اگر از جدول هش استفاده کرده باشیم، ابتدا مدیر پایگاه‌داده‌ها به جدول فوق مراجعه کرده و بر اساس نام و کلید یافته شده به آدرس حقیقی و مکان نام-کارمند در داخل جدول کارمند مراجعه خواهد کرد. طبیعی است که مراجعه به جدول فوق برای یافتن آدرس آن فرد سریع‌تر از پویش جدول و سطور است و دست‌یابی به اطلاعات سریع‌تر خواهد بود.

معایب فهرست درهم‌سازی در چیست؟

عیبی که برای جدول هش عنوان می‌کنند، مرتب نبودن اطلاعات در این روش است و همچنین تعداد زیادی از پرس‌وجو های مورد استفاده برای استفاده در پایگاه‌های داده وجود دارند که جداول هش کمکی در بهبود اجرای آنان ندارند. به عنوان نمونه، قصد دارید با استفاده از یک پرس‌وجوی ساده اس‌کیوال که اکثرا آن آشنایی دارید؛ به دنبال افرادی در کارمندان بگردید که سن آنان از ۴۰ سال کمتر باشد. در نظر بگیرید در جدول فیلدی با نام سن-کارمند (Employee Age) نیز وجود دارد. آیا خواهید توانست این پرس‌وجو را با استفاده از روش جدول هش انجام دهید؟ خوب، این کار ممکن نیست؛ زیرا جدول هش در مواقعی کاربرد خوبی از خود نشان خواهد داد که بخواهید به دنبال موارد کاملا مشخصی چون همانند رشته یا رقم خاصی باشید مثلا عنوانی آیا مساوی با عبارت خاصی است یا آیا شباهت به آن عبارت دارد. (مشابه این عبارت بعد از کوئری انتخاب مشابه زیر)

```
WHERE name = ‘Jesus’
```

این موضوع به این دلیل رخ می‌دهد که چیزی که در داخل کلید جدول هش عنوان شده‌است برای دسترسی به آن داده است و علاوه بر آن هیچ گونه نظم و ترتیب خاصی در چینش آنان در داخل فهرست جدول هش وجود ندارد. بنابراین اگر بخواهید بر اساس ترتیب فیلد سن-کارمند در جدول هش به دنبال کارمندان زیر ۴۰ سال باشید، به دلیل آن که ترتیبی در فهرست جدول هش وجود ندارد. نمی‌توانید بگوید این افراد را به صورت سریع برایتان بیابد؛ همین دلیل است که باعث شده که جدول هش به عنوان یک داده‌ساختار پیش‌فرض برای فهرست‌گذاری در سامانه‌های مدیریت پایگاه‌داده‌ها استفاده نشود. به طور کلی جداول هش کارایی و انعطافی که داده‌ساختار درخت بی ارائه می‌کند را نداشته و به خوبی آن نیست. یکی از مشکلات جدول هش تصادم است. در مثال قبل اگر کلید ویژه دو نام متفاوت از ده‌هزار نام اولیه‌مان یکی می‌شد باید کدام را در خانه مربوطه آرایه ذخیره می‌کردیم؟ به این اتفاق تصادم گفته می‌شود.

فهرست چگونه کارایی را افزایش می‌دهد؟

به این دلیل که فهرست اساسا یک ساختمان‌داده‌ها برای ذخیره مقادیر ستون‌های جدول است. جستوجو در این مقادیر بسیار سریع‌تر خواهد بود. همچنین اگر فهرست از ساختار داده درخت بی که یکی از داده‌ساختار معروف است استفاده کند، اطلاعات و مقادیر داخل جدول نیز به صورت مرتب شده خواهند بود که مزایای خاص خود را دارد. وجود اطلاعات در یک جدول به صورت مرتب‌شده مزایایی را به همراه دارد که همان‌طور که در قسمت معایب جدول هش ذکر شد. برای بهبود کارایی و عملکرد پایگاه‌داده‌ها در جستجو و ... تاثیر زیادی دارد.

حال بیایید تصور کنیم که در پایگاه‌داده‌ها خود یک فهرست با ساختمان‌داده‌ها درخت بی برای ستون نام-کارمند (Employee_Name) ساخته‌ایم. این به این معنی خواهد بود که اگر به دنبال کارمندی با نام عیسی «Jesus» گشتیم؛ این کار را توسط کوئری اس‌کیوال ساده‌ای که در اوایل مقاله نوشتیم انجام خواهیم داد. در این زمان برای جستوجو نیازی به پویش و جستجوی کل مقادیر برای یافتن نام کارمند مورد نظر «Jesus» نیست و در عوض پایگاه‌داده‌ها از فهرست برای یافتن کارمندی با نام عیسی «Jesus» استفاده خواهد کرد. به دلیل این که فهرست بر اساس حروف الفبا مرتب خواهد شد؛ جستوجو به دنبال فردی با نام عیسی بسیار سریع خواهد بود. فهرست فوق بر اساس حروف الفبا مرتب است و اگر به دنبال نام فردی مثلا عیسی «Jesus» بگردیم؛ این نام با حرف جی لاتین «J» آغاز شده است. در این زمان است که فهرست مرتب شده به کار می‌آید و به جای این که پایگاه‌داده‌ها در تمامی جدول به دنبال آن فرد باشد؛ تنها در آن افرادی جستوجو را انجام خواهد داد که حرف اول آنان جی «J» است. همچنین در داخل جداول اشاره‌گری را نیز در سطرهای جدول ذخیره می‌کند که باعث می‌شود بتوان مقادیر دیگر ستون‌ها را نیز بازیابی کرد.

به این صورت ذخیره کلیه مقادیر یک جدول به شکل مرتب‌شده در فهرست نیز کار عقل‌مدارانه‌ای نیست و همانند کپی و گرفتن پشتیبان از یک جدول است که باعث هدر

رفتن فضای ذخیره‌سازی و بزرگی بدون دلیلی پایگاه‌داده‌ها خواهد بود که برای مقادیر خیلی بزرگ بسیار غیربهینه و غیرمنطقی است. بنابراین همواره در فهرست ستونی که بیشتر به آن نیازداریم را قرار می‌دهیم. به عنوان نمونه برای سامانه اینترنتی یک بانک و پایگاه داده حساب‌های کارتهای اعتباری در آن سامانه، شماره کارت را در فهرست نگاه می‌داریم.

دقیقا چه چیزی در فهرست یک پایگاه‌داده‌ها وجود دارد؟

حال شما می‌دانید که فهرست پایگاه‌داده‌ها بر یک ستون از یک جدول ساخته می شود و در این فهرست مقادیر آن ستون خاص قرار گرفته‌است. اما این موضوع نیز مهم است که درک کنیم که فهرست مذکور به هیچ وجه اطلاعاتی که در دیگر ستون‌ها در آن جدول وجود دارد را نیز در خود نگاه نمی‌دارد. برای مثال اگر ما یک فهرست از نام-کارمندان (Employee_Name) ایجاد کنیم به این معنی است که سن-کارمند (Employee_Age) و آدرس-کارمند (Employee_Address) هم ستون‌های دیگر جدول کارمند هستند؛ در داخل فهرست پایگاه‌داده‌ها مذکور قرار ندارند.

یک اشاره‌گر نیز توسط فهرست در سطور جدول ساخته می‌شود. بنابراین سوال این است که مقداری که در جست‌وجوی آن هستیم در یک فهرست (مانند عیسی «Jesus») پیدا شود؛ مابقی فیلدهایی را که مرتبط با این فیلد یافت شده‌است را چگونه باید پیدا کنیم یا به آنان دست‌یابیم (فیلدهای دیگر مانند اطلاعات اطلاعات عیسی یا سن او)؟ خوب، موضوع کاملا ساده است؛ فهرست پایگاه‌های داده علاوه بر اطلاعات فیلد مورد نظر، از اشاره‌گرهایی را نیز در داخل فیلدهای مرتبط با این سطر نیز ذخیره کرده است. یک اشاره‌گر در واقع یک ارجاع است به مکانی از حافظه است که اطلاعات در دیسک ذخیره شده‌اند. پس علاوه بر مقادیر ستون که در فهرست ذخیره شده است؛ اشاره‌گری به آن سطر از جدول که آن ستون قرار دارد نیز وجد دارد که هرگاه مقداری را در فهرست یافتیم؛ به سطر مورد نظر نیز دسترسی خواهیم داشت. به عنوان نمونه در مثال مذکور و در کنار مقادیر گوناگون هر ستون مثلا کنار نام-کارمند در فهرست اشاره‌گری نیز به سطر مورد نظر قرار دارد. ("Jesus", 0x82829) که در این مثال عبارت «0x82829» به آن سطر و مکانی از حافظه که کارمند فوق در آن ذخیره شده است اشاره دارد. بدون این اشاره‌گرها شما فقط یک مشخصه را خواهید داشت که اگر بخواهید به مشخصات مرتبط با آن مشخصه دسترسی داشته باشی؛ با مشکل مواجه خواهید شد. مخصوصا این که اطلاعات فوق بر اساس ترتیب دیگری از جدول اصلی مرتب شده‌اند.

چگونه میفهمید پایگاه‌داده‌ها چه زمانی از یک فهرست استفاده می‌کند؟

زمانی که شما کوئری زیر را اجرا کنید؛ پایگاه‌داده‌ها بررسی می کند که آیا برای ستون مورد نیاز فهرستی پرس‌وجو شده‌است یا خیر. با فرض این که برای ستون نام-کارمند (Employee_Name) دارای یک فهرست باشد؛ پایگاه‌داده‌ها بررسی خواهد کرد که آیا استفاده از فهرست برای این پرس‌وجو مناسب باشد. در صورتی که پایگاه‌داده‌ها تصمیم‌گرفت که باید از فهرست استفاده شود، آنگاه فهرست مورد استفاده قرار خواهد گرفت. به‌خاطر این که در برخی حالت ممکن است که استفاده از فهرست معنی خاصی نداشته باشد و بهتر باشد از فهرست استفاده نکرد.

آیا می‌توان پایگاه‌داده‌ها را مجبور کرد در هر پرس‌وجو از فهرست استفاده کند؟

به طور کلی، شما نمی‌توانید به پایگاه داده بگویید که در چه زمان‌هایی از فهرست استفاده کند. این تصمیمات توسط خود پایگاه‌داده‌ها اتخاذ می‌شوند. اگر چه شایان ذکر است که در بسیاری از پایگاه‌داده‌ها (مانند اوراکل و MySQL) شما قادر خواهید بود که دقیقا مشخص کنید که پایگاه‌داده‌ها از فهرست استفاده کند.

روش ساخت فهرست توسط اس‌کیوال چگونه است؟

ساخت یک فهرست با استفاده از یک ستون؛ در زیر پرس‌وجوی «کوئری» اس‌کیوال را می‌بینید که برای ساخت یک فهرست برای ستونی با نام-کارمند (Employee_Name) در جدول کارمند است.

```
CREATE INDEX name_index ON Employee (Employee_Name)
```

ساخت یک فهرست با استفاده از چند ستون: اگر بخواهید دو یا چند ستون را در یک فهرست داشته‌باشد؛ باید پرس‌وجو «کوئری» زیر را اجرا کنید تا فهرست را اجرا نمایید. در مثال زیر ما از دو ستون استفاده کرده‌ایم؛ یکی ستون نام-کارمند (Employee_Name) و ستون دیگر سن-کارمند (Employee_Age) هستند.

```
CREATE INDEX name_index ON Employee (Employee_Name,
Employee_Age)
```

مثالی خوب برای برای فهم بیشتر مفهوم فهرست پایگاه‌داده‌ها؛ اگر بخواهیم قیاس خوبی برای تعریف فهرست در یک بانک‌اطلاعاتی عنوان کنیم، باید از فهرست در یک کتاب نام برد. فهرست‌ها در کتاب‌ها معمولا در اول کتاب آمده و از عناوینی که در کتاب وجود دارد به همراه شماره صفحه تشکیل شده‌است. در برخی کتب انگلیسی و حتی برخی کتب فارسی، فهرست مطالب در آخر کتاب قرار دارد. مثلا اگر شما کتابی را در مورد حیوانات مثلا سگ خریداری کنید؛ در این کتاب اطلاعات مختلف صفحات زیادی وجود دارد که به بخش و فصل‌های مختلف تقسیم شده‌اند. هر فصل و بخش نیز خود دارای عناوین مختلفی هستند که برای پیدا کردن آنان به فهرست مطالب رجوع می کنید. مثلا اگر به دنبال مطلبی درباره انواع نژاد سگ‌های شکاری باشید باید به فهرست مراجعه کرده و با مشاهده عنوان مرتبط با موضوع، به صفحه نوشته شده در برابر عنوان مراجعه و آن بخش را مطالعه کنید؛ این کار باعث صرفه‌جویی در وقت و افزایش سرعت دسترسی به مطلب خواهد بود. اگر کتاب دارای فهرست نبود؛ شما مجبور بودید تا تمامی مطالب کتاب را از اول پویش کنید تا به عنوان دلخواه دست‌یابید. در این زمان بهترین حالت برای پیدا کردن عنوان این بود که عنوان در اوایل کتاب باشد و بدترین حالت نیز زمانی است که عنوان در اواخر کتاب باشد.

فهرست در پایگاه‌داده‌ها نیز همین‌طور است و با استفاده از فهرست‌ها شما به مقادیر و ستون دلخواه خواهید رفت؛ بدون آن که وقت زیادی را صرف پویش کلیه اطلاعات نماید تا اطلاعات خود را پیدا کنید. فهرست در پایگاه‌داده‌ها تقریبا چنین کاری را انجام می‌دهد که فهرست کتاب انجام می‌دهد.

استفاده از فهرست چه مقدار هزینه دارد؟

خوب، همانند هر روش دیگری در دنیای واقعی، فهرست در پایگاه‌داده‌ها نیز معایب و مزایای خاص خود را دارد. یکی از معایب استفاده از فهرست همان‌طور که شاید تا الان پی برده باشید، لزوم استفاده از فضای بیشتر برای نگاهداری مقادیر در فهرست است که ممکن است فضای بیشتری برای پایگاه‌داده‌ها نیاز باشد. نکته دیگری هم که برای فهرست در پایگاه‌داده‌ها می‌تواند باعث کاهش کارایی شود؛ این است که در هر زمان که شما نیاز به درج، حذف و ویرایش اطلاعات در پایگاه اینترنتی یا جدول خاصی که دارای فهرست است را داشته باشید تغییراتی که در اطلاعات اعمال می‌شود باید در فهرست نیز اعمال شود؛ فراموش نکنید که اطلاعاتی که در فهرست ذخیره می‌شد با اطلاعات واقع در ستون مورد نظر برای فهرست باید مطابقت داشته باشند و در هر بار تغییر، تغییرات باید همگام شوند. بنا به قاعده‌ای کلی، هر فهرست می‌تواند فقط در زمانی که نیاز به پرس‌وجو داشتیم ایجاد شود.

موتورهای پایگاه‌داده در مای‌اس‌کیوال

تصور کنید که چه اتفاقی می‌افتاد اگر به عنوان یک راننده خودرو قادر بودید که براساس نیازهای مختلف که در سفر و یا استفاده از خودرو داشتید به یک گاراژ مراجعه کرده و موتور شما توسط یک مکانیک موتور ماشین خود با موتور دیگری تعویض می‌گردید. در این حالت شما برای هر شرایطی قادر بودید که از موتور دل خواه استفاده کنید؛ موتوری برای مسافرت، موتوری برای مسابقه و .... در مای‌اس‌کیوال دقیقا می‌توانید به چنین اقدام کرده و تعویض موتور در هر شرایط قابل انجام خواهد بود. به طور کلی تعویض موتور برای موتورهای پایگاه‌داده‌ها مورد استفاده توسط نرم‌افزار مدیریت پایگاه‌داده‌ها رابطه‌ای «RDBMS» مای‌اس‌کیوال یکی از مزایای مهم آن به شمار می‌رود. به این صورت که هر وقت نیاز داشتید، فراخور نیاز خود بتوانید به تغییر و تعویض موتور پایگاه‌داده‌ها به مورد دلخواه اقدام کنید.

قطعا ممکن است همان موتور پیش‌فرضی که در مای‌اس‌کیوال مورد استفاده است برای شما کافی باشد؛ با این حال در برخی شرایط استفاده از موتورهایی که برای آن شرایط خاص عملکرد بهتری را به نمایش می‌گزارند، به بازدهی بهتری منجر خواهد شد. در برخی از مواقع حتی ممکن است نیاز داشته باشید با استفاده از دانش برنامه‌نویسی و رابط برنامه‌نویسی نرم‌افزار «API» مای‌اس‌کیوال خود دست به ایجاد یک موتور سفارشی بزنید. یکی از ویژگی‌های نرم‌افزارهای آزاد / متن‌بازنیز همین امکان سفارشی‌سازی بالای آنان با نگاه به کدهای دیگر است که باعث بهبود کارایی نرم‌افزار به دلیل قابلیت سفارشی‌سازی و انطباق بالا خواهد بود. در این مورد نیز به راحتی می‌توانید موتور خود را با انشعاب و سفارشی کردن یکی از موتورهای فعلی ایجاد کنید.

در نرم‌افزار مدیریت پایگاه‌داده‌های رابطه‌ای مای‌اس‌کیوال تقریبا اکثر موتورهای پایگاه‌داده‌های موجود برای مای‌اس‌کیوال را می‌توان استفاده کرد؛ به شرط آن‌که در زمان کامپایل نرم‌افزار آن‌ها فعال بوده و غیرفعال نشده باشند؛ معمولا در اکثر توزیع‌های گنو/لینوکس و یا نسخه موجود در داخل پایگاه اینترنتی اوراکل از نسخه‌ای استفاده شده است که از اکثر موتورهای مرسوم اس‌کیوال استفاده می‌کند. با این حال اگر در توزیع شما از مای‌اس‌کیوال استفاده شده است که دارای موتور مورد نیاز شما نیست می‌توانید از نسخه موجود در پایگاه اینترنتی اوراکل استفاده کرده و یا آن را خودتان از منبع کامپایل و نصب کنید.

کامپایل نرم‌افزار از پایه شاید برای توسعه‌دهندگان سیستم‌عامل ویندوز کمی ناآشنا باشد، اما این موضوع برای توسعه‌دهندگانی که از سیستم‌عامل‌های شبه یونیکس استفاده می‌کنند، کاملا موضوعی عادی به شمار می‌آید.

مای‌اس‌کیوال از سه موتور پایگاه داده استفاده می‌کند که آی‌زم «ISAM» مای‌آی‌زم «MyISAM» و هیپ «HEAP» نام دارند؛ دو نوع موتور پایگاه‌داده‌ها دیگر برای پایگاه‌های داده نیز با نام‌های اینودی‌بی «InnoDB» و برکلی «Berkley BDB» نیز از دیگر موتورهای پایگاه داده در مای‌اس‌کیوال به شمار می‌آیند.

آی‌زم «ISAM»

موتور آی‌زم یک موتور پایگاه‌داده‌ها است که روش مناسب و سریع برای مدیریت جداول در پایگاه‌داده‌ها است. این موتور با این ایده طراحی شده است که معمولا در اکثر پایگاه‌های داده در بیشتر مواقع به جای آن‌که نیاز به به‌روز کردن ویرایش اطلاعات باشد؛ نیاز به مشاهده و اجرای پرس‌وجو احساس می‌شود. بنابراین این موتور برای استفاده در مای‌اس‌کیوال از سرعت بالایی در خواندن اطلاعات جداول دارد و به راحتی می‌تواند بدون درگیر کردن منابع سیستم در حافظ اصلی و جانبی، اطلاعات را به راحتی نمایش دهد. با وجود این، دو نکته منفی درباره این موتور را می‌توان به عدم پشتیبانی از تراکنش (transactions) اشاره داشت یا همچنین نکته منفی مهم دیگر به عدم تحمل خطا در این موتور اشاره داشت. برای مثال اگر دیسک سخت شما از کار بیفتد و یا با مشکل مواجه شود، موتور دیگر نمی‌تواند که اطلاعات را بازیابی کرده و آنان را مجدد احیا کند. اگر از این موتور در کاربردی تجاری با امکان خطرات بالا استفاده می‌کنید؛ بهتر است از پایگاه داده خود پشتیبانی زنده و ۲۴ ساعته داشته باشید که انجام این کار با استفاده از برخی ویژگی‌های موجود در مای‌اس‌کیوال به راحتی قابل انجام است.

مای‌آی‌زم «My ISAM»

مای‌آی‌زم را می‌توان نسخه سفارشی مای‌اس‌کیوال از موتور آی‌زم دانست

کـه موتور پیش‌فرض در نرم‌افزار مدیریت پایگاه داده رابطه‌ای مای‌اس‌کیوال اسـت. بـه علاوه فراهم کـردن امکاناتی برای فهرسـت‌گذاری و مدیریت فیلد کـه در آی‌زم وجـود نداشـته اند. مای‌آی‌زم از یک شـیوه قفـل کردن جدول «table locking» بهره می‌برد که باعث بهبود کارایی بیشتر در زمان خواندن و نوشـتن‌های همزمان در پایگاه‌داده خواهد بود. نکته منفی برای این موتور این اسـت که در اکثر مواقع نیاز دارید که دسـتور بهینه‌سـازی جدول را در اکثر مواقع اجرا کنید؛ تا فضاهای از دست رفته توسط الگوریتم به‌روزرسانی را از بازیابـی کنـد. مای‌آی‌زم همچنین از چند یک ابزار افزونه «Extensions» مفید نیز برخوردار است؛ مانند افزونه «MyISAMChk» که وظیفه تعمیر فایل پایگاه‌داده‌ها را به عهده داشته و یا «MyISAMPack» که ابزاری برای فضاهای از دست‌رفته است.

موتور بر روی سـرعت بالا تاکید داشـته و عملکرد خوبی را در این زمینه بر جا گذاشـته اسـت و عملیات خواندن در آن بسـیار سریع است. این به همان دلیل است که نرم‌افزار مدیریت پایگاه‌داده‌ها رابطه‌ای مای‌اس‌کیوال در اکثر پایگاه‌هـای اینترنتـی مورد اسـتفاده اسـت و بـه یکی از سـریع‌ترین مدیران پایگاه‌داده‌ها در بسـتر اینترنت تبدیل شـده است. سـرعت خواندن اطلاعات و نمایـش اطلاعـات در پایگاه‌های اینترنتی از اهمیت بالایی برخوردار اسـت زیرا که اکثر عملیاتی که برای اسـتفاده از پایگاه‌داده‌ها در اینترنت اسـتفاده می‌شود؛ معطوف به خواندن اطلاعات از جداول پایگاه‌داده‌ها است و به همین دلیل هم اکثر IPPها کاربران را مجبور به استفاده از این موتور کرده‌اند.

هیپ «HEAP»

موتـور هیـپ اجازه می‌دهـد جداول موقتـی در داخل حافظـه اصلی اقامت داشـته باشـند. اقامـت در حافظه اصلـی باعث بهبود و افزایش سـرعت عمل موتـور خواهـد بود و سـرعت ایـن موتـور را از موتورهـای آی‌زم و مای‌آی‌زم بیشـتر خواهـد کـرد. بـا این حـال این نـوع ذخیـره داده در حافظـه جانبی بسـیار فرار اسـت و اگر اطلاعات قبل از این‌که سیسـتم خاموش شوند، هنوز در حافظـه اصلـی باشـند؛ اطلاعـات فوق به راحتـی از دسـت خواهند رفت. هیـپ عـلاوه بر ایـن از هدر رفتن فضـای خالی در زمانی که برخی سـطرها پـاک می‌شـوند؛ جلوگیـری می‌کند. همچنین جـداول هیـپ در مواقعی که می‌خواهیـد انتخاب‌هایـی تودرتـو برای انتخاب و دسـتکاری داده‌ها داشـته باشـید؛ گزینـه مناسـبی به شـمار می‌آید. لازم اسـت یادآوری کنـم که بعد از این‌کـه کارتـان با جدول تمام شـد آن را «destroy» کنید؛ بگذارید مجددا تاکید کنم که هرگز فراموش نکنید؛ هرگاه کار شما با جدولی تمام شد؛ آن را «destroy» کنید.

اینودی‌بی «InnoDB» بِرْکلْی دی‌بی «Berkley»

موتورهـای اینودی‌بـی و بِرْکلْـی دی‌بـی را می‌تـوان موتورهایـی دانسـت که محصولات مسـتقیم فناوریی هسـتند که باعث انعطاف مای‌اس‌کیوال و رابط برنامه‌نویسـی نرم‌افزار مای‌اس‌کیوال++ شـده‌اند. تقریبا تمامی چالش‌هایی را که در اسـتفاده از مای‌اس‌کیوال داشـته‌اید؛ مسـتقیما از این حقیقت ناشـی می‌شـود کـه موتـور پایگاه‌داده‌ها آی‌زم و مـای‌آی‌زم در تراکنش و کلیدهای خارجـی مشـکلاتی را داراسـت. در هر حال پشـتیبانی از ایـن دو موضوع در این دو موتور بسیار بهتر از سه موتور معرفی شده است و اگر این دو قابلیت برای شـما مهم هسـتند، باید از این دو موتور پایگاه‌داده‌ها استفاده کنید. در اکثر مواقع از موتور اینودی‌بی بیشـتر از موتور پایگاه‌داده‌ها برکلی اسـتفاده می‌شود. با وجود این سرعت در هنگام خواندن اطلاعات در دو موتور مذکور [اینودی‌بی «InnoDB» بِرْکلْی دی‌بی «Berkley»] بسیار پایین‌تر از سه موتور معرفی شده دیگر است.

بـا ایـن حال اگـر توانایی بیشـتری را در خود احسـاس می‌کنیـد؛ می‌توانید موتورپایگاه‌داده‌های مختص به خود را نیز با اسـتفاده از رابط برنامه‌نویسـی نرم‌افزار مای‌اس‌کیوال++ نوشـته و از آن اسـتفاده کنید. رابط برنامه‌نویسـی نرم‌افـزار مذکور تمامی مواردی را که برای سـاخت یـک موتور پایگاه‌داده‌ها، دسترسی به سطور، فیلدها، جداول، پایگاه‌های داده‌ها، ارتباطات، حساب‌های امنیتـی و دیگـر موارد مورد نیـاز و توابع پایگاه‌داده‌ها را کـه نیاز دارید را در اختیار شما قرار خواهد داد. آموزش استفاده از این رابط برنامه‌نویسی نرم‌افزار در این نوشته امکان‌پذیر نیست؛ با این حال در همین حد اشاره خواهم کرد که رابط برنامه‌نویسـی نرم‌افزار مای‌اس‌کیوال++ برای توسعه‌دهندگان وجود دارد و همیـن رابـط برنامه‌نویسـی نرم‌افـزار اسـت که پشـت قابلیت تعویض موتـور در مای‌اس‌کیـوال اسـت و ایـن قابلیـت را مدیـون مای‌اس‌کیـوال++ هسـتیم. بـا این حال این سـبک از موتور های قابل نصـب در مای‌اس‌کیوال می‌تواند به موارد دیگری مانند سـاخت یک فراهم کننده اکس‌ام‌ال بومی در مای‌اس‌کیوال نیز استفاده شود.

تعویض موتور، تمامی انعطاف موجود را توسـط یک افزونه مای‌اس‌کیوال که برای انسـی اس‌کیوال نوشـته شده اسـت؛ فراهم آورده است. این کار توسط پارامتر نوع «TYPE» انجام می‌شود. همچنین مای‌اس‌کیوال به شما این اجازه را نیـز خواهـد داد کـه بـرای هر یـک از جداول مختلـف، موتور پایـگاه داده خاصـی را اختصـاص دهید. در برخی از مواقع بـه آنان قالب جدول «table format» نیز اطلاق می‌شـود. مثالی که در زیر نوشـته شده‌است؛ یک کوئری سـاده اس‌کیوال برای سـاخت چند جدول اسـت که برای هریک از آنان نوع جدول به انواع مختلف تغییر یافته است. به ترتیب انواع به کار رفته در مثال زیر موتورهای مای‌آی‌زم، آی‌زم و هیپ هسـتند. گفتنی است تمامی جداول سـاخته شـده از فیلدهای یکسـانی برخوردارند؛ با این تفاوت که نام آن‌ها به همراه نوع آن‌ها با هم تفاوت دارد.

```
CREATE TABLE tblMyISAM (
        id INT NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (id),
        value_a TINYINT
) TYPE=MyISAM
CREATE TABLE tblISAM (
        id INT NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (id),
        value_a TINYINT
) TYPE=ISAM
CREATE TABLE tblHeap (
        id INT NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (id),
        value_a TINYINT
) TYPE=Heap
```

همچنین می‌توانید از «ALTER TABLE» برای انتقال یک جدول که از موتور خاصی استفاده می‌کند به جدول دیگر با موتورپایگاه‌داده‌های دیگر استفاده کنید. برای مثال در کد نوشته شده اس‌کیوال زیر از دستور «ALTER TABLE» برای انتقال جدول به جدول دیگر استفاده شده است. اولین جدول از موتور مای‌آی‌زم استفاده می‌کند که با دستور زیر نوع «TYPE» آن به اینودی‌بی تغییر خواهد یافت که به معنای تغییر موتور در جدول مقصد است.

```
ALTER TABLE tblMyISAM CHANGE TYPE=InnoDB
```

مای‌اس‌کیوال این کار را در سه مرحله انجام خواهد داد. در اولین مرحله یک رونوشت از جدول مبدأ تهیه خواهد کرد. در مرحله بعدی تمامی تغییرات وارد شده در یک صف قرار می گیرند تا این‌که عملیات انتقال از یک موتور به موتور دیگر به طور کامل انجام شود. سپس در مرحله بعدی تغییراتی که در صف قرار داشتند یک به یک بر روی جدول مقصد اعمال شده و عملیات انتقال با موفقیت به پایان خواهد رسید.

نکته، راهی میانبر برای «ALTER TABLE»؛ اگر می‌خواهید به شکلی ساده جدول خود را از یک جدول بهره‌مند از آی‌زم به جدولی بهره‌مند از مای‌آی‌زم منتقل کنید؛ به راحتی قادر خواهید بود تا از دستوری کوتاه‌تر که روشی میانبر است استفاده کنید.

```
mysql_convert_table_format
```

برای این‌که مشاهده کنید که یک جدول خاص از چه موتور پایگاه داده‌ای استفاده می‌کند؛ می‌توانید از دستور «SHOW TABLE» استفاده کنید. با استفاده از این دستور در داخل کد انسی اس‌کیوال به راحتی قادر به تعیین نوع موتور استفاده شده در هر جدول خواهید بود. خروجی دستور فوق یک جدول است که اطلاعات مختلفی را از یک جدول به نمایش در می‌آورد؛ با این حال آن قسمت از اطلاعاتی که موتور مورد استفاده در جدول اشاره دارد؛ اطلاعاتی است که در بخش نوع «TYPE» قرار دارد.

```
SHOW TABLE STATUS FROM tblInnoDB
```

نکته، راهی جایگزین برای نمایش اطلاعات جدول؛ به جای استفاده از دستور بالا و نوشتن دستور «SHOW TABLE»، می‌توانید از دستور زیر استفاده کنید؛ که اطلاعاتی مشابه با دستور قبلی را نمایش خواهد داد.

```
SHOW CREATE TABLE [TableName]
```

در انتها؛ چیزی بد تر از این نیست که موتور مورد نظر شما در مای‌اس‌کیوال وجود نداشته باشد و در همراه آن نباشد؛ همانطور که ذکر شد برای استفاده از یک موتور باید همراه با مای‌اس‌کیوال در زمان کامپایل، کامپایل شود تا بتوان از آن استفاده کرد. با وجود این اگر موتور فوق به همراه مای‌اس‌کیوال مورد استفاده شما نباشد و بخواهید از آن موتور در یک جدول استفاده کنید؛ پیغام خطایی برای شما نمایش داده نخواهد شد تا متوجه عدم وجد آن موتور در مای‌اس‌کیوال شود. اگر مای‌اس‌کیوال موتور مورد نظر را پیدا

نکند، از موتور پیش‌فرض مای‌آی‌زم استفاده خواهد کرد که برای کسانی که حتما به موتور خاص نیاز دارند اصلا خوب نیست. این در حالی است که ممکن است در آینده چنین قابلیتی در مای‌اس‌کیوال افزوده شود و برنامه‌ریزیهایی برای درج پیغام هشدار در هنگام نیافتن موتور مورد نظر در مای‌اس‌کیوال در ذهن توسعه‌دهنده‌های مای‌اس‌کیوال وجود دارد تا این مشکل را در آینده حل کنند. با این وجود و در حال حاضر تنها کاری که از شما بر می‌آید آن است که با استفاده از دستور بالا مشاهده کنید که جدول فوق از چه موتوری استفاده می‌کند. اگر از یک میزبان اینترنتی استفاده می‌کنید؛ از آن شرکتی که خدمت فوق را دریافت کرده‌اید و از قسمت پشتیبان حتما نصب بودن موتور دلخواه خود را جویا شوید.

انتخاب بیشتر به معنی بازدهی بیشتر خواهد بود؛ با وجود این‌که در مای‌اس‌کیوال از یک موتور خوب به صورت پیش‌فرض استفاده شده است که از سرعت و بازدهی تقریبا مناسب برخوردار است؛ چه نیازی به استفاده و تعویض موتور پایگاه داده به دیگر موارد وجود خواهد داشت؟ همچنین به وجود این‌که موارد مختلف جهت استفاده به عنوان موتور در پایگاه داده عرضه شده‌است؛ چرا باید موتور اختصاصی ساخته و استفاده کنیم؟ جواب سوالات فوق بسیار ساده است. نکته مهم در این است که هر گاه بتوانید انتخاب بیشتری داشته باشید و دستتان در ساخت ابزاری سفارشی مورد نظر خودتان باز باشد؛ آنگاه موتور فوق بهترین تطابق را با نیازهای شما خواهد داشت.

قطعا مای‌آی‌زم موتوری سریع برای پایگاه داده به حساب می‌آید. با این حال اگر شما از قابلیت انتقال «transactional» در پایگاه داده استفاده کرده و به آن نیاز دارید؛ حتما باید از دو موتور پایگاه داده‌ای که از این قابلیت پشتیبانی دارند؛ استفاده کنید. با این حال با استفاده از قابلیتی در مای‌اس‌کیوال که به شما اجازه استفاده از موتوری خاص را تنها بر روی یک یا چند جدول خاص می‌دهد؛ شما به راحتی خواهید توانست از یکی از موتورهای پایگاه داده برکلی یا اینودی‌بی برای آن جدول که به قابلیت مذکور نیاز دارد استفاده کرده و برای دیگر جداول از موتور پیش‌فرض مای‌آی‌زم استفاده کنید تا سرعت بیشتری را برای نمایش اطلاعات به همراه پشتیبانی از انتقال «transactional» را در پایگاه داده را نیز داشته باشید. به صورت کلی نکته مهم و کلیدی در نرم‌افزار مدیریت پایگاه داده رابطه‌ای «RDBMS» مای‌اس‌کیوال را می‌توان انعطاف بالای آن دانست.■

ارتباط پایگاه‌داده‌ها با دنیای خارج

نویسنده:
وحید سهرابلو

در ۲۰۰۳ مفهومی به SQL اضافه شد بام نام SQL/MED که هدف آن ارتباط پایگاه‌داده‌ها با خارج از دنیای خود بود. این مفهوم در PgSQL با نام FDW مخفف Foreign data wrappers است معرفی شد. FDW تقریبا امکان ارتباط با هر Wrapper خارجی را می‌دهد از DBMS های مختلف گرفته تا NoSQL ها و حتی فایل‌های CSV و git و ...

یک مثال ساده

```
-- load extension first time after install
CREATE EXTENSION mongo_fdw;

-- create server object
CREATE SERVER mongo_server FOREIGN DATA WRAPPER mongo_fdw
OPTIONS (address '127.0.0.1', port '27017');

-- create foreign table
CREATE FOREIGN TABLE customer_reviews
(
   _id NAME,
   customer_id TEXT,
   review_date TIMESTAMP,
   review_rating INTEGER,
   product_id CHAR(10),
   product_title TEXT,
   product_group TEXT,
   product_category TEXT,
   similar_product_ids CHAR(10)[]
)
SERVER mongo_server
OPTIONS (database 'test', collection 'customer_reviews');

-- collect data distribution statistics
ANALYZE customer_reviews;
```

در این مثال یک جدول با نام customer_reviews ایجاد می‌شود که اطلاعات خود را از سرور mongo دریافت می کند.

این جدول‌ها قابلیت انجام تراکنش بین سرورها ، Sort ، Join و حتی Aggregate را داراست. البته توقع کارایی مناسب در همه وارپرها را نداشته باشید. بسته به وارپری که استفاده می‌شود و روش‌های ارتباطی با مقصد می‌تواند کارایی متفاوتی را داشته باشد.

Wrapper های بسیار زیادی برای PgSQL نوشته شده است. از وب‌سرویس‌های SOAP گرفته تا git و تویتر و ... . اما اگر شما نیاز به نوشتن یک Wrapper جدید داشته باشید باید دانش نوشتن اکستنشن به زبان C را داشته باشید و صد البته حوصله نوشتن با زبان C. اگر شما هم مثل من هیچ علاقه‌ای به نوشتن با C ندارید می‌توانید از یک اکستنشن با نام multicorn استفاده کنید.

این اکستنش به شما امکان می‌دهد که Warpper را با زبان Python پیاده‌سازی کنید. نوشتن Warpper کار سختی نیست پیشنهاد می کنم به صفحه http://multicorn.org مراجعه کنید

FDW FUN

```
DROP EXTENSION IF EXISTS www_fdw CASCADE;
CREATE EXTENSION www_fdw;
CREATE SERVER www_fdw_server_google_search FOREIGN DATA WRAPPER
www_fdw
   OPTIONS (uri 'https://ajax.googleapis.com/ajax/services/search/web?v=1.0');
CREATE USER MAPPING FOR current_user SERVER www_fdw_server_goog-
le_search;
CREATE FOREIGN TABLE www_fdw_google_search (
   /* parameters used in request */
   q text,

   /* fields in response */
   GsearchResultClass text,
   unescapedUrl text,
   url text,
   visibleUrl text,
   cacheUrl text,
   title text,
   titleNoFormatting text,
   content text
) SERVER www_fdw_server_google_search;

postgres=# select url,substring(title,1,25)||'..',substring(content,1,25)||'..'
   from www_fdw_google_search where q='postgres';
            url              |        ?column?         |        ?column?
-----------------------------------+-------------------------+-------------------------
http://www.postgresql.org/          | <b>PostgreSQL</b>: The wo... | Sophis-
ticated open-source...
http://www.postgresql.org/download/   | <b>PostgreSQL</b>: Downlo... |
The core of the <b>Postgr...
http://www.postgresql.org/docs/       | <b>PostgreSQL</b>: Docume... |
There is a wealth of <b>P...
http://en.wikipedia.org/wiki/PostgreSQL | <b>PostgreSQL</b> - Wikip... |
<b>PostgreSQL</b>, often ...
```

۱۹۶۰:

اولین مدل‌های کامپیوتری شده پایگاه داده

۱۹۷۰:

مدل رابطه ای بعنوان برنده‌ی اصلی در بین مدل‌های دیگر اعلام می‌شود

عصر طلوع پایگاه داده

- مدل رابطه‌ای و زبان برنامه نویسی آن یعنی SQL ادغام میشوند
- مدل مخرب باعث نابودی مدل‌های دیگر می‌شود

۱۹۸۰:

موفقیت های تجاری پایگاه داده‌ی رابطه‌ای

یک توسعه ی تجاری

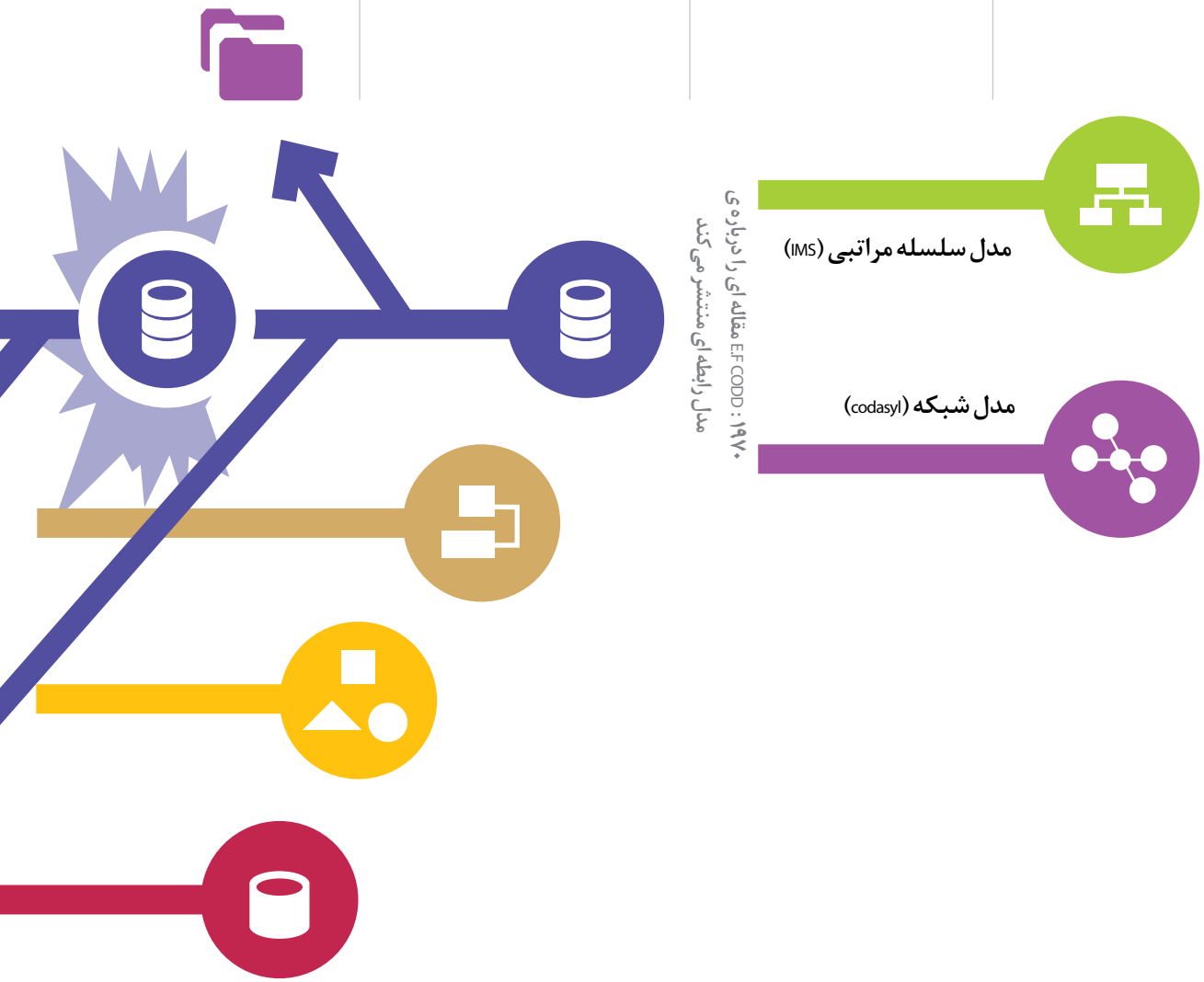
- SQL به یک استاندارد بالفعل بدل می‌شود
- پیشنهادهای تجاری از طرف IBM، رشد بازار اوراکل
- ورود مدل‌های دیگر به صحنه ی بازار، البته با جلب توجه نه چندان زیاد

۱۹۹۰:

پایگاه داده‌ی رابطه ای غرق می‌شود

تغییرات در تکنولوژیج

- انفجار داده ها با آمدن عصر اینترنت
- پایگاه های داده ی SQL که با یک سرور کار می کنند، با مشکل منابع مواجه می‌شوند
- حوزه‌های هوش تجاری و تجزیه و تحلیل داده، از پایگاه داده‌های معاملاتی خارج می‌شوند



۱۹۹۰:

SQL توزیع شده و NOSQL ادغام می‌شوند

بازیگران جدید به هم ملحق می‌شوند

- پایگاه های داده‌ی جدید معاملاتی SQL معرفی شده اند
- پایگاه داده‌های NOSQL خلاء موجود در پردازش داده های سازمان نیافته را پر می کنند
- Hadoop با تجزیه و تحلیل پتابایت‌های(Petabyte : معادل یک میلیون گیگا بایت) داده توجه‌ها را جلب می کند

امروز:

پایگاه های داده‌ی Scale-out کمی جلب توجه می کنند

پایگاه‌های داده با علم روز منطبق میشوند و رشد می کنند

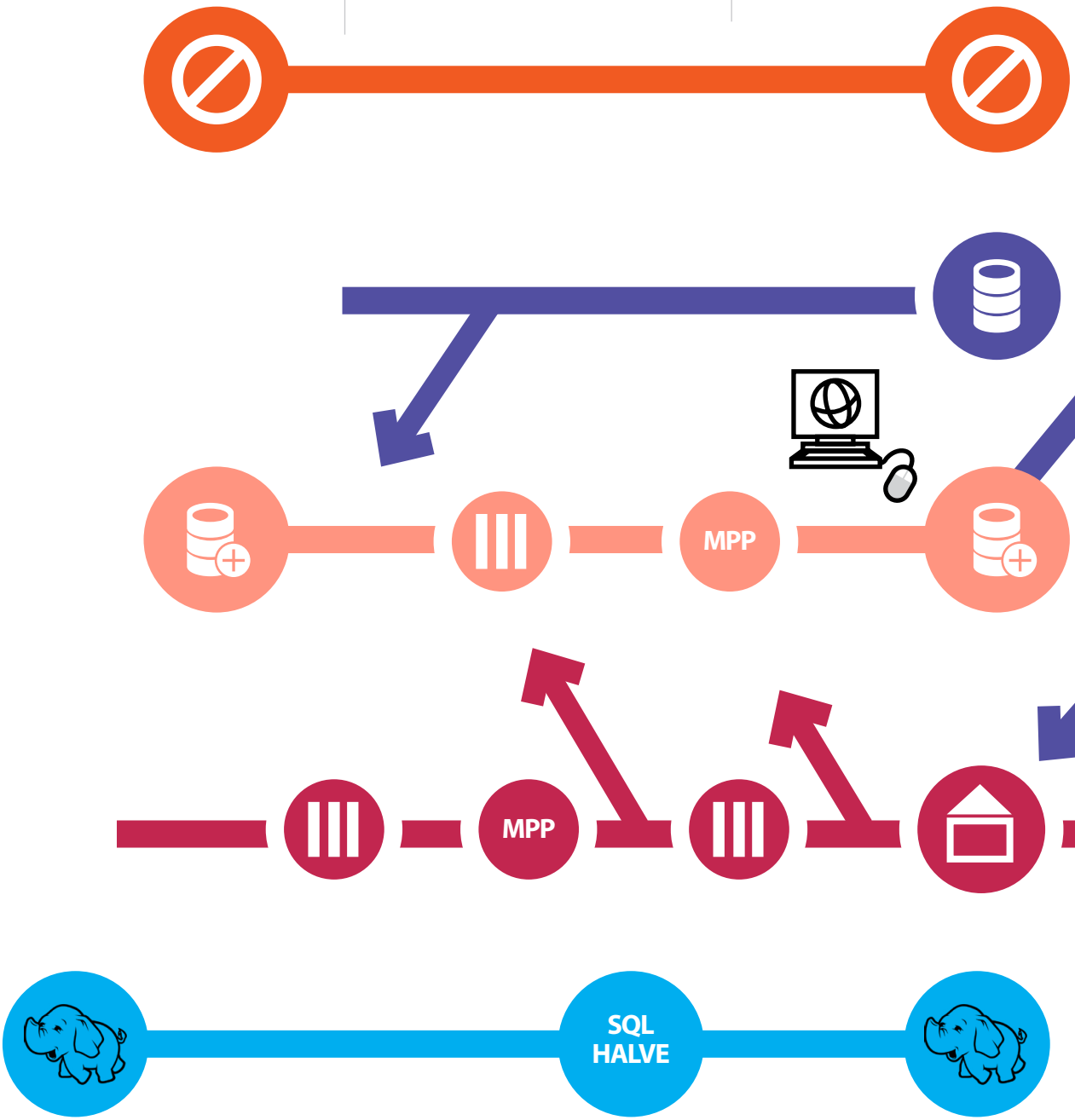
- کسب و کارها خواهان تجزیه و تحلیل های لحظه ای بر روی داده‌های عملیاتی هستند
- Scale-up SQL ثابت می کند که پر هزینه است، اما Scale-out SQL مشکلات محدودیت منابع را رفع میکند
- Scale-out تجزیه و تحلیل‌های آنی را همراه با مقادیر بالای نقل وانتقالات فراهم می کند
- Google و Clustrix در این فضا پیشگام هستند

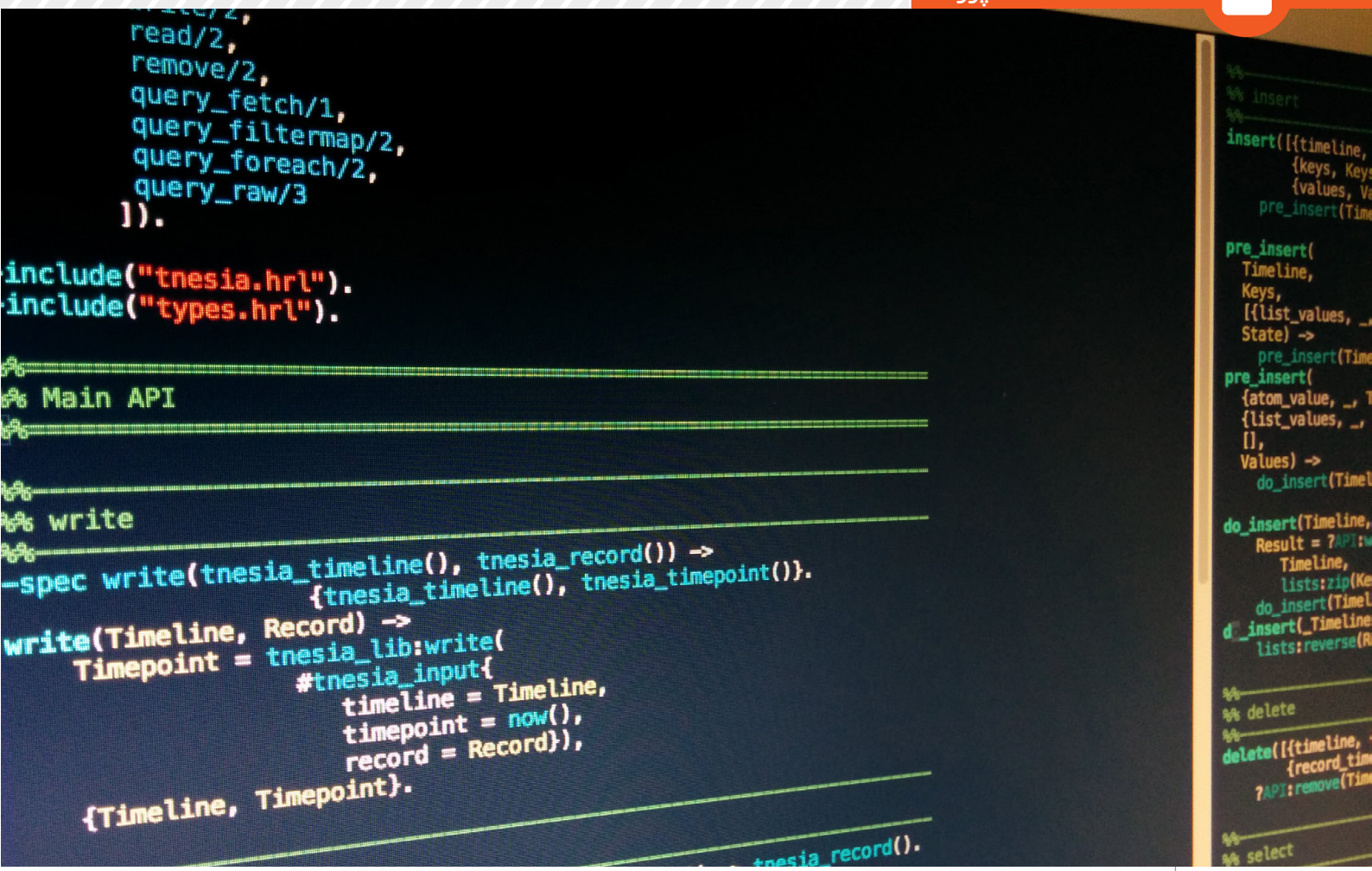
آینده:

میراث Scale-up جایگزین شده است

کسب و کارها با نوآوری‌ها در حوزه ی پایگاه داده پیشرفت می‌کنند

- Single Node SQL جای خود را به Scale-out SQL میدهد
- تجزیه و تحلیل های انواع داده های انبار شده در پایگاه داده‌های آنی قابل دسترس می‌شوند
- کسب و کارها تحرک و چابکی فراوانی میگیرند





سرگذشت یک پروژه متن باز

چرایی، چیستی و چگونگی بانک اطلاعاتی Tnesia

نویسنده: حمیدرضا سلیمانی

در این مقاله به بررسی بانک اطلاعاتی Tnesia میپردازیم و به این سوالات پاسخ میدهیم: Tnesia چیست، چرا به وجود آمده و چگونه میتوان از آن استفاده کرد.

چرایی

پاییز سال ۹۳ وقتی قرار شد ویژگی جدیدی به نام کانال به نرم افزار بیسفون اضافه کنیم، من به عنوان توسعه دهنده سمت سرور مسئول طراحی و پیاده سازی موضوع شدم. کانال ویژگی ای بود که موسسات، نشریات و در کل تولید کنندگان محتوا میتوانستند از طریق آن برای مخاطبانشان که کاربر بیسفون بودند محتوا تولید کنند و با آن‌ها در تماس باشند.

در مراحل طراحیه اولیه همه چیز خوب پیش رفت تا آنجایی که نوبت به انتخاب بانک اطلاعاتی برای نگهداری محتوای کانال ها شد. ویژگی بارز این داده ها زماندار بودن آن‌ها بود، به صورتی که کاربرانی که آن‌ها را دنبال می‌کنند هر بار که آنلاین می‌شوند بتوانند از آخرین زمانی که آنلاین بوده اند داده های جدید را بازیابی کنند. در برآوردهای اولیه فهمیدیم که با توجه به تعداد کاربران جاری و تعداد کانال های پیش بنی شده، در ساعات پیک شاید تعداد درخواست های بازیابی محتوای کانال ها به چند میلیون درخواست در ثانیه برسد. با این فرض نیاز به یک بانک اطلاعاتی مناسب کاملا احساس شد. اولین گزینه‌هایی که روی میز گذاشته شد Riak، InfluxDB، Postgresql و Cassandra بودند که همه آن‌ها خوشبختانه متن باز بودند.

ویژگی بارز این داده ها زماندار بودن آن‌ها بود، به صورتی که کاربرانی که آن‌ها را دنبال می‌کنند هر بار که آنلاین می‌شوند

بانک اطلاعاتی Postgresql با توجه به رابطه ای بودنش برای نگهداری داده های زماندار بسیار مناسب بود، اما برای توزیع پذیر کردن آن با هدف داشتن یک سیستم Highly Avail-able چالش های زیادی پیش رو داشتیم که باعث شد در همان ابتدای امر از آن صرف نظر کنیم.

بانک اطلاعاتی InfluxDB هم از این جهت که

به صورت کلی مخصوص نگهداری داده های زماندار بود گزینه مناسبی بود، از طرف دیگر بـا دارا بودن ویژگی توزیـع پذیری به عنوان یکی از خصوصیات اصلی، نگرانی ما را نسبت به داشتن یک بانک اطلاعاتی Highly Availa-ble از بین میبرد. اما چیزی که باعث شد در نهایت از آن استفاده نکنیم API آن بود که بر روی HTTP پیاده سـازی شـده بود. از انجایی که از طرفی HTTP یک Text Protocol است و در مقایسه با Binary Protocol ها هزینه زیادی صـرف Parse کـردن آن میشـود و همچنین حجـم داده هـا را هـم بالا میبـرد، و از طرف دیگر نیازمند برقراری یک کانکشن جدید به ازای هر درخواست است، تصمیم گرفتیم به سراغ گزینه بعدی برویم.

بانـک اطلاعاتـی Riak هـم بـا دارا بـودن ویژگی‌هـای مناسـب توزیـع پذیـری مانند Highly Available، Partition Tolerancy و Eventual Consistency گزینه بسـیار مناسبی بود اما از انجایی که درخشـش این ویژگی‌ها زمانی میسر بود که از آن به عنوان یک بانک اطلاعاتی Key/Value استفاده شود، مجبور به حذف این گزینه هم شدیم.

در نهایـت بانـک اطلاعاتـی Cassandra را انتخاب کردیم که از لحاظ توزیع پذیری تمام ویژگی‌های Riak را دارا بود به علاوه سـاختار Column Family آن که کاملا مناسب بود برای مـدل کردن داده های زماندار. از طرف دیگر با داشتن یک زبان جستجوگر به نام CQL که شـباهت هایی به SQL داشـت و همچنین به صـورت Binary هم بـود در رقابت با InfluxDB برنده میشد.

ایـن انتخـاب زمانی متزلزل شـد که متوجه شـدیم Client Driver های موجود Cassandra بـرای زبـان Erlang کـه زبـان اصلی مـا برای توسـعه نرم افزار بود قابل اطمینان نیستند، زیـرا یـا به روز نبودند و یـا تمام ویژگی های لازم را نداشـتند و از همه بدتر هر کدام تنها یک توسعه دهنده داشتند که اگر متاسفانه بر اثر تصادف رانندگی فوت میکرد، ما به مشکل جدی ای برمیخوردیم.

اینجا بود که تصمیم گرفتم این مشکل را به نحو دیگری حل کنم و پیشنهاد پیاده سازی یک بانک اطلاعاتی کوچک با حداقل امکاناتی که به آن‌ها نیاز داریم را به مدیریت مجموعه دادم. خوشـبختانه با موضوع موافقت شـد و مراحل طراحی و پیاده سـازی Tnesia شروع شـد. پس از آزمایش‌های اولیه متوجه شدم کـه نه تنها میتوان از آن بـرای نیاز خودمان استفاده کنیم، بلکه انتشار آن به صورت متن باز باعث بالا رفتن انگیزه توسـعه آن خواهد شد زیرا میتوان از حمایت جامعه متن باز به صورت های مختلف استفاده کرد و همچنین تبلیغی موثر نیز برای مجموعه خواهد شد.

در نهایـت موافقـت شـرکت با انتشـار آن به صورت متن باز باعث این مهم شد.

چیستی

در ابتـدا Tnesia تنهـا یـک الگوریتـم Time-Series Indexing بر روی بانک اطلاعات Mnesia بـود، امـا پس از گذشـت زمـان و با اضافـه کـردن ویژگی‌هـای مختلـف بـه آن دیگـر میتـوان بـه آن یـک بانـک اطلاعاتی گفت. مثل هر بانک اطلاعاتی دیگری، هدف Tnesia ذخیره اطلاعات است، اما با تمرکز بر روی داده های زماندار. داده های زماندار، داده هایی هسـتند که بر اساس زمان رخدادشان ذخیره و بازیابی میشوند. این داده ها میتوانند در فواصل زمانی معین باشند مانند داده هایی که سنسورهای هواشناسی ذخیره میکنند، و یا با فواصل زمانی متغییر ذخیره شوند مانند لاگ هـای یـک وب سـرور و یا حتی توییت های شما در توییتر.

با همین هدف، Tnesia با معرفی سـه مفهوم Timestep، Timeline و Timepoint داده هـای زماندار را دسـته بنـدی میکند. به این صـورت کـه کاربـر آن میتوانـد با بـه وجود آوردن Timeline هـای متفـاوت هر داده خود را بـا یـک Timepoint که همـان زمان جاری اسـت در Timeline مـورد نظـر ذخیـره کند. ایـن Timepoint هـا در Timestep هایـی بـا فواصل زمانی یکسان ذخیره می‌شوند. وجود Timestep کمک میکند که در هنگام بازیابی داده ها بین دو فاصله زمانی، به جای جستجو تمام Timeline تنها قسـمت‌هایی که احتمال وجود داده مورد نظر وجود دارد بررسی شود. همچنین با پیش بینی میـزان تراکم داده و تغییر Timestep در تنظیمات Tnesia میتوان سرعت بازیابی داده‌ها را حتی بهبود بخشید.

چگونگی

بـرای اسـتفاده از Tnesia دو راه وجـود دارد. راه اول اسـتفاده آن بـه عنـوان یـک Dependency در نـرم افـزاری که بـا Erlang نوشـته شده اسـت. در این راه شما میتوانید بـا اسـتفاده از API فراهـم شـده به زبـان Er-lang داده هـای خـود را بـا سـرعت بالایـی ذخیـره و بازیابـی کنیـد. راه دوم نصـب Tnesia به صورت Stand-Alone اسـت که در آن نیازی به استفاده از زبان Erlang برا ذخیره و بازیابی اطلاعات ندارید و میتوانید با استفاده از زبان جستجوگر TQL بر روی HTTP به بانک اطلاعاتی خود دسترسی داشته باشید. البته راه دوم سرعت کمتری نسبت به راه اول دارد به خاطر اسـتفاده از یک زبان جستجو گر و همچنین هزینه‌های پروتکل HTTP.توضیحات بیشتر راجع به چگونگی استفاده از Tnesia را می‌توانید در صفحه گیت هاب آن به آدرس https://github.com/bisphone/Tnesia مطالعه کنید

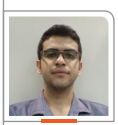
بررسی سرگذشت پروژه های متن باز بیان گر این موضوع است که حتی پروژه های متن باز بزرگی که ما همه آن‌ها رامیشناسیم و از آن‌ها استفاده‌میکنیم از پروژه‌های کوچکی شروع شده اند که هیچوقت تصور آن را هم نمی‌کردند که روزی به این شهرت و کاربرد برسند

جمع بندی

بررسی سرگذشت پروژه های متن باز بیان گر این موضوع اسـت که حتی پروژه های متن باز بزرگی که ما همه آن‌ها را می‌شناسیم و از آن‌ها استفاده می‌کنیم از پروژه‌های کوچکی شروع شده اند که هیچوقت تصور آن را هم نمی‌کردند که روزی به این شهرت و کاربرد برسند. از این رو شاید پروژه‌های کوچک من و شما روزی به چنین مرحله برسند، تنها اگر برای توسعه آن وقت بگذاریم و به فکر انتشار آن‌ها به صورت متن باز بیفتیم.

??? ?????? ?? ??? ?? ?????
AES-128 ??????? ?????
??? ?????? ????? ??? ?? ??? ??? ????? ??? ???
? ???

نکاتی که در فارسی باید رعایت کنیم



نویسنده:
مهدی خشنودی

هنگام کار با پایگاه‌داده‌های مختلف، معمولا در ابتدای آشنایی برای ذخیره و بازیابی داده‌های فارسی، با مشکلات عجیب و قریبی مواجه می‌شویم. در این مقاله چند نکته ساده را برای رفع چنین مشکلاتی عنوان می‌کنیم.

ارتباط اولیه

حتما یادتان هست اولین روزهای برنامه‌نویسی را که قصد اتصال به پایگاه‌داده‌ها را داشتید. گاهی اطلاعات به صورت علامت سوال ذخیره می‌شود یا حتی اطلاعاتی که به درستی ذخیره شده، به صورت علامت سوال یا نویسه‌های ناخوانا در می‌آید. ظاهرا همه چیز خوب درست انجام شده. پس مشکل کجاست؟

این مساله جزو اولین مواردی است که باید درنظر داشته باشید و به سادگی نیز برطرف می‌شود؛ تنها با فراخوانی دستور زیر بعد از برقراری اتصال:

```
set names 'utf8';
```

یکسان‌سازی

بارها اتفاق افتاده است، هنگامی که الکی مثلا جستجویی در داده‌ها انجام می‌دهیم، نتیجه‌ای را برنمی‌گرداند. به پایگاه‌داده‌ها می‌گوییم: «خودم فلان اطلاعات را وارد کردم. چرا پیداش نمی‌کنی؟» گیر کار کجاست؟

البته خوشبختانه ما آزادکاران که از مزایای گنو/لینوکس بهره می‌جوییم، چنین گیرهای مسخره‌ای را نداریم و این نانی است که مایکروسافت و به طبع اون شرکت‌های توسعه‌دهنده محصولات مایکروسافتی در دامان ما (یا آنها) گذاشتند. در گنو/لینوکس صفحه‌کلید استاندارد فارسی فعال است و به همین خاطر معمولا همه چیز درست کار می‌کند. اما در زمان‌های قدیم که مایکروسافت فارسی را پشتیبانی نمی‌کرد، مجبور بودیم که از صفحه‌کلید عربی استفاده کنیم و تقریبا این عادت سالیان سال است که همراه کاربران ایران شده و به خصوص بخش دولتی، هنوز از این عادت بدون این که خیلی آگاه باشند، رنج می‌برد.

ما در فارسی حرف «ی» یا «ک» نداریم اما چون هنوز برخی صفحه‌کلیدها این حروف را به صورت عربی وارد می‌کنند، ممکن است به دنبال «یک» باشید و غافل از این که در پایگاه‌داده‌ها «یک» یا «یک» با «ی» عربی یا حتی «یک» ذخیره شده باشد. پس باید هنگام ذخیره یا واکشی داده‌ها، همواره یک منش را پی بگیریم که قطعا نویسه‌های فارسی در اولویت است.

ویژگی فارسی پایگاه‌داده‌ها، جداول و فیلدها

گاهی هنگام ذخیره‌سازی، حروف در جداول به صورت ناخوانا ذخیره می‌شوند حتی با این که نکته اول را رعایت کردید. گاهی هم هنگام مرتب کردن، برخی رکوردها به انتهای جدول می‌رود. چرا؟

علت آن است که یکی از خصیصه‌های پایگاه‌داده‌ها یا جداول یا فیلدها را بر روی «utf8-persian-ci» تنظیم نکرده‌اید و عباراتی که با «ک»، «گ» و «ژ» شروع می‌شود را در انتها یا ابتدا می‌آورد. شماره یونیکد این حروف بعد از حروف عربی است.

ذخیره تاریخ

برای ذخیره‌ی تاریخ (به خصوص تاریخ خورشیدی) در پایگاه‌داده‌ها روش‌های مختلفی وجود دارد.

ذخیره‌ی تاریخ به صورت timestamp: در این روش یک تاریخ خاص (معمولا unix time، یک ژانویه ۱۹۷۰) را به عنوان مبدا در نظر گرفته و تعداد ثانیه‌های سپری شده تا تاریخ مورد نظر را ذخیره می‌کنند. (در این روش عدد به دست آماده برای تاریخ شمسی و میلادی یکسان بوده) استفاده از نوع داده‌ی datetime: در این روش به سادگی می‌توان تاریخ مورد نظر را با این نوع داده ذخیره کرد. البته برای تاریخ شمسی ابتدا باید تاریخ را به میلادی تبدیل کرده و سپس آن را با این نوع داده ذخیره کنید. در نظر داشته باشید که گاهی نوع داده‌ی datetimeoffset که همیشه یک منطقه زمانی را نیز با خود حمل می‌کند می‌تواند بسیار مفید واقع شود. ذخیره‌ی تاریخ به صورت رشته‌ای: در این روش تاریخ را به صورت یک رشته ذخیره می‌کنند. برای مثال '01.04.2015'. در این روش همیشه اعداد روز و ماه را به صورت دو تایی ذخیره کنید تا در سرعت مرتب‌سازی و جستجو بهینه‌تر عمل کنید. اما توصیه می‌شود هرگز تاریخ (شمسی) را به صورت متن ذخیره نکنید تا بتوانید از توابع بهترین استفاده‌ها را داشته باشید. استثنائا انگلیسی را پاس بداریم در نهایت هم توصیه می‌شود که در نام‌گذاری پایگاه‌داده‌ها، جداول و فیلدها از فینگلیش استفاده نکنید، مثل پسر/دختر خوب معادل انگلیسی آن را بیاورید و تا حدامکان کوتاه و در عین حال با معنا باشد. باور کنید انتخاب نام درست خودش هنره. البته می‌تونید فارسی هم استفاده کنید اما با توجه به احتمال بروز مشکلات بعدی، بهتر است فراموشش کنید. اما اگر اصرار دارید، باید هنگام صدا زدن نام را در میان نویسه «`» قرار دهید، همانند زمان استفاده از اسامی، دارای فاصله بین کلمات.

سه محدودیت NoSQL

نویسنده:
وارن کرنوچان

برای سال‌ها جایگزین‌های SQL مبتنی بر هادوپ مورد توجه کاربران بود، این جایگزین‌ها ارزان و متن‌باز بودند و به خوبی در محیط ابرهای عمومی تجربه شده بودند. در بسیاری از موارد این پایگاه‌داده‌ها قابلیت مقیاس‌پذیری برای حجم گسترده‌ای از «بزرگ داده» را در مقابل انواع رابطه‌ای عظیم الجثه داشتند. البتـه باید درنظر داشـت کـه پردازش‌کننده پایگاه‌داده‌های NoSQL برای بدسـت آوردن این پیروزی، یک رویکرد متفاوت را نسبت به انواع رابطه‌ای/SQL در مدیریت داده‌ها، انتخاب کردند. رویکردی که به طور ویژه آن‌ها را در اسـتفاده، نسـبت به پایگاه داده‌های SQL بسیار محدودتر می‌کرد.

سه مورد از این محدودیت‌ها عبارتند از:

- عدم انسجام داده مکرر
- عدم پشتیبانی تجزیه و تحلیل «عمیق» تطبیقی
- عدم داشتن ویژگی دسترسی‌پذیری بالا برای ابرهای خصوصی و ترکیبی
- عدم انسجام داده مکرر

هادوپ، مپ‌ردیوس، مانگو دی‌بی، هایو و غیره طراحی شده‌اند تا فراتر از پایگاه‌های داده SQL در ابرهای عمومی با ورودی بسـیار سـریع از داده، قابل مقیاس‌پذیری باشـند. بـرای ایـن هدف بـه آن‌ها امکان «سـازگاری شـرطی» در هـر عملیات «commit» اجازه داده شـده اسـت که باعث اطمینان یافتن از سازگاری در زمان انجام عملیات با تاخیر می‌شود. مزیت این روش جلوگیری از نگه داشتن تراکنش‌ها (transaction) اسـت. البته در طول این تاخیر، برخی داده‌ها نادرست خواهند بود که کاربر این مساله را به عنوان هزینه پرداختی برای بینش اضافی که از داده‌ها پیدا می‌کند، می‌پذیرد.

اخیرا سیستم‌های برپایه هادوپ شروع به اداره کردن عملیات به‌روزرسانی سریع در ابرها کرده‌اند. در بیشتر مواقع سیستم‌های برپایه هادوپ، داده‌ها را قبل از آن که بر روی دیسـک نوشـته شوند، آماده می‌کنند. در نتیجه اگر سیستم پیش از نوشـتن داده ها بر روی دیسـک «کرش» کند، داده‌ها از بین می‌روند. پس از این اتفاق اسـت که تجزیه و تحلیل‌ها با یک مجموعه از داده‌ها روبرو می‌شـوند که با داده‌های خوانده شده اصلی متفاوت‌اند.

در این سـال‌ها پایگاه‌داده‌های مبتنی بر هـادوپ، مانند مانگو دی‌بی، درصدی از داده‌ها را که سـازگارند، بهبود بخشـیده‌اند. اما هنوز هم به خوبی پایگاه داده‌های رابطه‌ای نیستند.

علاوه بر این، پایگاه داده‌های مبتنی بر هادوپ شروع به ارائه کردن «دسترس‌پذیری بـالا» عملیات‌هـا کرده‌اند که باعث اطمینان از نوشـتن بر روی دیسـک پیش از خواندن می‌شود. البته در بسیاری از موارد هنوز هم به خوانده شدن قبل از نوشته شدن، اجازه می‌دهند.

عدم پشتیبانی تجزیه و تحلیل «عمیق» تطبیقی

پایگاه‌داده‌های مبتنی بر هادوپ، برای دست یافتن به کارایی بالا، از سربار نرم‌افزاری معناشناسی‌داده‌ها، به شکل پیچیده جلوگیری می‌کنند. این نوع معناشناسی به کاربر نهایی این امکان را می‌دهد تا «what-if» را با استفاده از تجزیه و تحلیل کردن چند بعد از ویژگی‌های داده به وسیله روشی انعطاف‌پذیر انجام دهند.

راه‌هـای فراوانـی برای قطعه کـردن داده وجود دارد و یا این کـه داده‌ها در نوع و معنا نسـبتا سـریع در حال تغییر هستند. انجام دادن چنین «پرسشی (query)» در لحظه ممکن است ترجیح داده شود، اما وقتی دانشمندان داده‌ها می‌خواهند تا با بررسی عمقی، بیشتر در مورد چیزی که اتفاق می‌افتد بدانند، SQL و رابطه پیروز می‌شود.

عدم داشـتن ویژگی دسترسـی‌پذیری بالا بـرای ابرهای خصوصی و ترکیبی

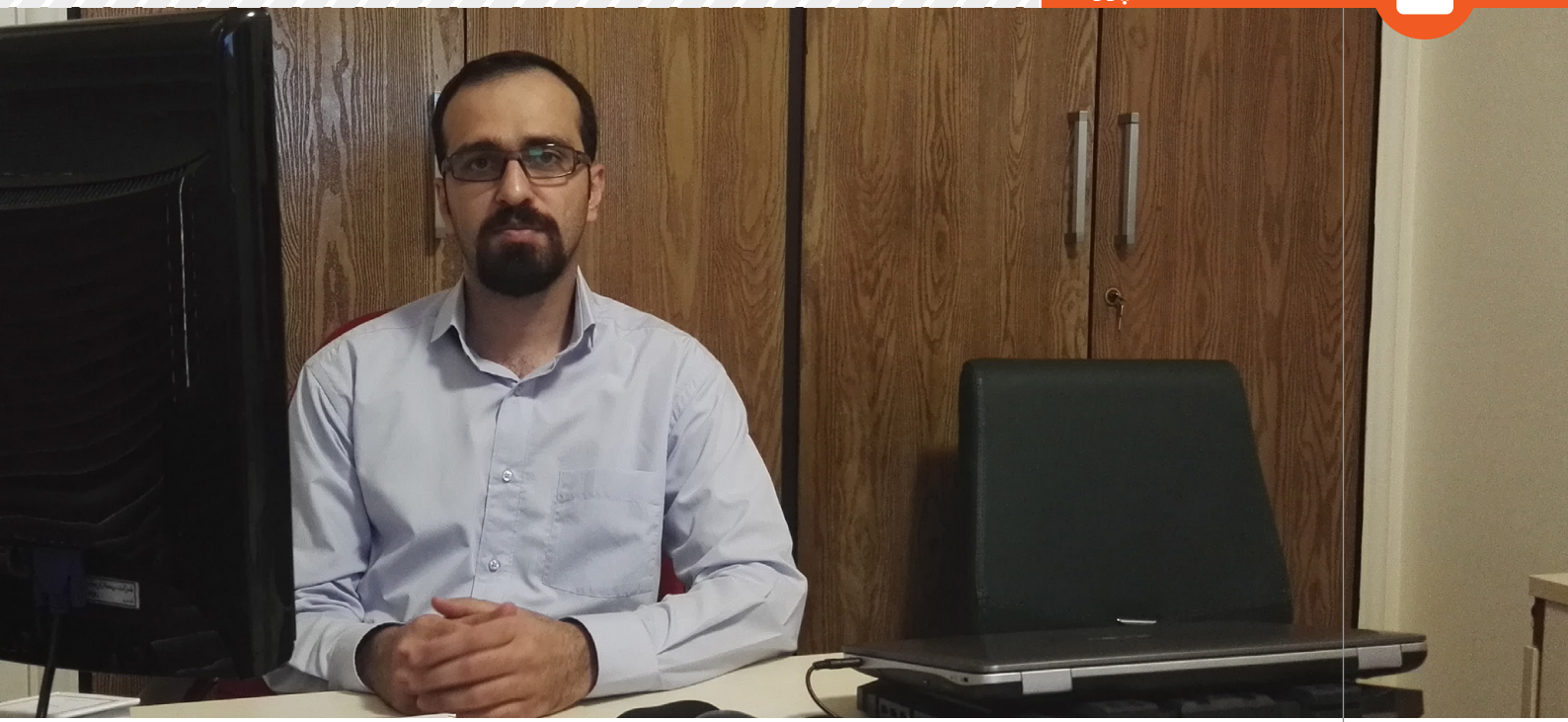
ایـن موضـوع تقریبـا زمانی کـه NoSQL تـازه ظهور کرده بـود دیده می‌شـد، اما محدودیت بزرگی اسـت. در حالی که Oracle، EMC، IBM یا یک SAP ممکن اسـت بسیاری از ویژگی‌های remote-copy، failover و roll-back/roll-forward را ارائه کنند، ایـن ویژگی‌هـا در NoSQLها تقریبا نادر اسـت. علاوه بر ایـن پایگاه‌داده‌های SQL سال‌های بسیار متمادی زمان داشته‌اند تا بهتر و بهتر راهی را برای بدست آوردن «دسترس‌پذیری بالا» در محیط‌های مقیاس بزرگ و مقیاس متوسط، مشخص کنند. اما راه‌کارهای NoSQL در ابرهای عمومی مقیاس بزرگ انبوه به دنیا آمده‌اند و اکنون باید خود را با ابرهای خصوصی تطبیق بدهد.

SQL یا NoSQL یا چیزی دیگر

یکی از سخنرانان نشست «In-Memory Computing Summit» که در سانفراسیسکو برگزار شد و در آن هر دو نوع پایگاه‌داده‌های SQL و NoSQL به خوبی معرفی شدند، گفت: آن‌ها یک جابجایی را در کلاینت‌هایشان از SQL به NoSQL دیده‌اند و اکنون در حال مشاهده بازگشت دوباره به سمت SQL هستند. معنی این اتفاق به گوشه رانده شدن NoSQl نیست. اما می‌توان حدس زد که NoSQL بیشتر و بیشتر بر روی پـردازش «عملیاتی» داده تمرکز کرده اسـت در حالی که پایگاه‌داده‌های SQL به نمایش قدرتشان در تجزیه و تحلیل ادامه می‌دهند.

بنابراین محدودیت‌های NoSQL نشان‌دهنده یک تقسیم کار است که باعث ایجاد یک حس بزرگ در بسیاری از کاربردها با حتی یک معماری پایگاه‌داده‌های ابری وسیع یا سازمانی وسیع می‌شود.

محدودیت‌های NoSQL به معنای فقدام سودمندی به شکل کلی نیست و آن‌ها از پس وظایف بر می‌آیند. با این حال حدس زده می‌شود کاربران زرنگ هواخواهی خودشان را برای هم نوعان هادوپ محدود خواهند کرد.■



به بهانه بررسی یکی از بزرگ‌ترین پایگاه‌داده‌ها، گلوگاهی کشف شد

دردطـــــراحی

رضا فیوجی از سال ۸۳ در تیم مهاجرت داده پردازی بوده و بعد از آن به قول خودش با همان تفکر آی‌بی‌امی به بهسازان ملت آمد. اصلی‌ترین مشتری‌های بهسازان، بانک ملت و وزارت بازرگانی است که مدتی می‌شود در حال راه‌اندازی سامانه تدارکات برای الکترونیکی کردن خریدهای دولت است. در چارت سازمانی این شرکت، زیرمجموعه مدیرعامل چندین معاونت وجود دارد و بخش عملیات شامل زیر بخش‌هایی مانند پشتیبانی، بهینه‌سازی، ادمین حضوری و ادمین غیرحضوری و... است.

با توجه به این‌که دیشب تا دیر وقت بیدار بودید و در شرکت کار می‌کردید معمولا چه کارهایی را در نیمه شب انجام می‌دهید؟

کارهای پر ریسکی که سیستم را با اختلال مواجه می‌کند. مثلا وقتی می‌خواهیم نسخه را عوض کنیم سیستم داون می‌شود و باید از آن یک پشتیبان داشته باشید و روال انتقال سرور پشتیبان را رعایت بکنید. یعنی به درستی کابل عوض کنید و روی آن همه چیز شبیه این طرف باشد که کم‌ترین زمان عدم پاسخگویی را داشته باشیم بعد نسخه را نصب کنیم و تستش کنیم و دوباره در محیط عملیاتی تست می‌شود تا مطمئن شویم برای کاربر واقعی هم درست کار می‌کند. یک سایت ساده نیست کلی هزینه کردیم روی محصولات IBM سرورهای لینوکسی نصب کردیم. پروژه‌های تحت جاوا روی این‌ها سوار می‌شود.

چقدر پیش می‌آید این تغییر نسخه اتفاق بیفتد؟

این یکی از تغییرات ماست تغییرات دیگری مثل به‌روزرسانی و تغییر گذرواژه و به‌روزرسانی سخت‌افزار هم هست. یکسری تغییرات هم ناخواسته و اجباری است مثل مشکلات دیتاسنتر و از کار افتادن شبکه بانک. کار که بزرگ می‌شود مشکلات هم تصاعدی زیاد می‌شود. عمده سیستم‌های غیرحضوری پیش ما شاپرک و بانکداری اینترنتی بانک ملت و همراه بانک ملت است. جمعا با سرورهای پشتیبان ۳۵۰ تا سرور داریم.

عمده سیستم‌های غیرحضوری پیش ما شاپرک و بانکداری اینترنتی بانک ملت و همراه بانک ملت است. جمعا با سرورهای پشتیبان ۳۵۰ تا سرور داریم.

روی هر سرور دیتابیس جدا دارید؟

نه. ما این‌جا چند شغله‌ایم. ادمین لینوکس و محصولات آی‌بی‌ام هستیم این بخش عمده کار ماست و پایگاه‌داده سهم کم‌تری در شغل ما دارد. تقریبا ۴۰ - ۵۰ سرور دیتابیسی دارم و بقیه‌اش وب اپلیکیشن و وب سرور یا لینوکس ادمین انجام می‌دهم که در لایه بیزینس لاجیک یا لایه اپلیکیشن است. یک اپلیکیشن جاوایی روی ساختار آی‌بی‌ام نصب می‌شود. این‌ها به پایگاه‌داده‌های اصلی کُربانکینگ ما وصل می‌شوند که آی‌بی‌امی هستند. یکی از معاونت‌های شرکت DBA است.

در بخش عملیات یکسری پایگاه‌داده‌های لوکال است که ارزش نداشته این‌ها را برایش مین‌فریم هزینه کنند و دیتابیس‌های کوچکتر و میانی که DB2 تحت PC هستند. ارزش و اهمیتشان خیلی فاصله‌اش کم است با آن چیزی که روی مین‌فریم است. چون در داده‌کاوی و وب‌کاوی به آن‌ها احتیاج داریم و مراجع قانونی به آن نیاز دارند. ما چندتا مین‌فریم داریم مثل سامانه مکفا. پنج سال پیش یک مین‌فریم را کلا خالی کردیم و DBA هم شدیم قبلش این‌قدر جدی نبود MySQL و SQL server داشتیم.

این‌جا از چه نرم‌افزارها و ابزارهای آزادی استفاده می‌کنید؟

MySQL یک دانه بیشتر نداریم. ما نرم‌افزار متن‌باز کم داریم همان سوزه اینترپرایز هم که استفاده می‌کنیم متن‌باز نیست چون مشتری ما را مجبور کرده است که از چه سیستمی استفاده کنیم. شرکت مکفا که به همراه بانک ما کمک می‌کند می‌گوید من یک MySQL آن طرف دارم تو هم باید یک MySQL داشته باشی وگرنه استاندارد این‌جا لینوکس برای سیستم‌عامل است، WebSDL آی‌بی‌ام برای وب‌سرور است و DB2 برای پایگاه‌داده است. تنها ابزار آزادی که من استفاده می‌کنم nagios است که یک مانیتورینگ جامع برای همه ابزارهای تحت لینوکس و شبکه است. تقریبا ۲۵ آبجکت را من الان دارم مانیتور می‌کنم. رم، اکسپشن، هاردها، CPU و تعداد کاربرها را روی سرورها مانیتور می‌کنیم و واقعا عالی است. حتی تنظیم کردیم در چه مواقعی برای ما پیامک بیاید. خوبی‌اش این است که می‌توانیم برایش پلاگین هم بنویسیم.

مشکلاتی که برای پایگاه‌داده‌ها پیش می‌آید و کندی آن‌ها معمولا به چه دلیل است؟

طراحی یکی از مشکلات عمده است و چیزی است که اصلا دیده نمی‌شود به نظرم طراح‌ها هکر هستند. چون یک چیز غلط طراحی می‌کنند و فردا کندی دیتابیس و شبکه و اپلیکیشن رو اول مشتری می‌بینند بعد ادمین و تولید و بعد می‌گویند آقای طراح تو هم یک نگاهی بکن درحالی که هکر آی‌تی طراح‌ها هستند که با طراحی درست می‌توانند کاری کنند که هیچ‌کس ناراحتی نداشته باشد.

به نظر شما نگاهی که به استفاده از پایگاه‌های داده در ایران هست با کشورهای دیگر متفاوت است؟

آن چیزی که من شنیدم به لحاظ فنی هیچ‌کس به اندازه ایران متخصص نیست. چون ما شغل مشخصی نداریم همه‌مان همه کاره هستیم و مجبوریم به خاطر این‌که همه درست کار نمی‌کنند وارد کار همه بشویم و سر از خیلی چیزها دربیاوریم. من از کنترل پروژه گرفته تا دیتاسنتر را بلدم. در کشورهای صنعتی چون هرکس کار خودش را انجام می‌دهد این‌طور نیست. به لحاظ تکنیکال ما از خیلی‌ها قوی‌تریم برای همین راحت اپلای می‌کنیم. تمام نمره منفی‌هایی که من به همکارانم می‌دهم بابت سواد نیست بابت عدم دقت و تعامل و هماهنگی است.

چیزهایی که ما نمی‌دونیم طراحی درست و تشخیص و اندازه یک بیزینس است. اندازه کار را تشخیص نمی‌دهیم و می‌خواهیم این چیزی که بلد هستیم را ارائه دهیم و پولش را بگیریم. شرکت ما چون IBM بیس بوده و همه ما در IBM سابق و داده‌پردازی کار کردیم داریم IBMای زندگی می‌کنیم. البته به درد بیزینس بانک می‌خورد اما خیلی جاها این طوری نیست

وضع DBAها در ایران و جهان را چطور ارزیابی می‌کنید؟

در ایران بهتر است کسی که می‌خواهد در این زمینه کار کند یک بعدی نباشد. مثلا می‌روند اوراکل یاد می‌گیرند. در حالی که بسیاری با یاد گرفتن یک ابزار کار خود را تمام شده فرض می‌کنند. مثل این‌که فقط بگویی من این کار را با انبردست باز می‌کنم. تو دیتابیس کار نشدی خیلی هنر کنی اوراکل کار شدی. در این زمینه اگر می‌خواهیم کاره‌ای بشویم باید طراح دیتابیس شویم و اصول پایگاه‌داده را بدانیم. ابزار اصلا مهم نیست. بیزینس ایران را نگاه می‌کنی و آن‌ها را یاد می‌گیری. یاد گرفتن که کاری ندارد نوع استفاده از ابزارها و دیدت نسبت به کار مهم است که یک بحث جداست. من در مصاحبه‌هایم یک کلمه فنی از کسی سؤال نمی‌کنم. رزومه رو ببینم و تشخیص بدهم به درد ما می‌خورد فقط با او صحبت می‌کنم. هیچ سؤال فنی نمی‌پرسم. ما ابزار کار نباید باشیم. یعنی با سوادترین فرد را می‌آوریم این‌جا احساس بی‌سوادی می‌کند. برای همین سامانه بانکی شب عید می‌افتد. چرا؟ مگر مین‌فریم زد آی‌بی‌ام یا اوراکل نداریم؟ خود ما مشکلمان طراحی است.

وضعیت آموزش دیتابیس در ایران چطور است؟

ما کلا در ایران آموزش نداریم همه باید خودشان سعی کنند وقتی جایی می‌روند کار بگیرند. ما حتی در شرکت خودمان هم آموزش کاملی نداریم. چون روند استخدام طول می‌کشد می‌گویم لینوکس را خودت بخوان چون کم‌ترین کار من لینوکس است. تقریبا سه ماه هیچ کاری با نفر جدید نداریم جز این‌که شیفت بیاید پیش بقیه باشد و کارهای دیگران را ببیند و یاد بگیرد و اصلا برایم مهم نیست بعد آزمایشی بخواهد با رزومه سه برابر از این‌جا برود. چون هیچ سودی برای من ندارد که کار را ذره ذره به او بدهم. اگر طرز فکرش این است که برود هیچ جایی کاری نمی‌تواند یاد بگیرد. هیچ بخل و خستی نباید در آموزش انجام دهی. شما می‌دانید شغل server administrator اصلا در سازمان فناوری اطلاعات تعریف نشده. تنها شغل administrator که تعریف شده ادمین شبکه است.

به اعتقاد شما نباید این‌ها را قبلش در دانشگاه بیاموزند؟

اینکه قطعا درست است. اما خیلی شغل‌ها را نمی‌توانی؛ چون خاص هستند و ابزارها در شرکت به نوع خاصی استفاده می‌شود. یعنی من لینوکس را طور خاصی در این‌جا استفاده می‌کنم و بیزینس این‌جا را باید یاد کار آموز بدهم. هرکسی که این‌جا می‌آید باید یکسری ابزار بلد باشد و سازمان را بشناسد و بداند از ابزارها در سازمان چطور استفاده کند. درسته که بیرون از این‌جا هیچ آموزش درست و هدف‌مندی وجود ندارد باید به کارآموز فرصت دهید که خودش یاد بگیرد. شما بعد از دانشگاه تازه باید میلیونی هزینه کنی و یاد بگیری. کسی که آموزش می‌دهد هم طرف را تعصبی بار می‌آورد. فکر می‌کند اگر اوراکل بداند در دیتابیس سرهنگ تمام است. در حالی که اصلا دیتابیس کار نیست. هیچ‌جا نمی‌گویند مشکل آموزش ما تعصبی بار آوردن افراد است. حالا متن‌باز تفکر دموکراسی‌مآبانه‌ای دارد و دوستانه‌تر است و من از این دفاع می‌کنم ولی وقتی وارد جزئیات می‌شوی خیلی بد است بگوییم فقط سوزه یا فقط DB2. تو فقط می‌توانی بگویی تجربه‌ات در یک زمینه بیشتر است.

اگر بخواهیم دیتابیس‌های موجود را برای کارهای مختلف دسته‌بندی کنید چطور این کار را می‌کنید؟

قطعا در فعالیت‌های بانکی که با تراکنش‌های مالی سر و کار دارند باید از ابزار مهمی مثل DB2 آی‌بی‌ام در بستر مین‌فریم استفاده کنیم. چون از اسکیل بزرگ می‌تواند پشتیبانی کند و تمام

> مثلا می‌روند اوراکل یاد می‌گیرند. در حالی که بسیاری با یاد گرفتن یک ابزار کار خود را تمام شده فرض می‌کنند. مثل این‌که فقط بگویی من این کار را با انبردست باز می‌کنم. تو دیتابیس کار نشدی خیلی هنر کنی اوراکل کار شدی. در این زمینه اگر می‌خواهیم کاره‌ای بشویم باید طراح دیتابیس شویم و اصول پایگاه‌داده را بدانیم

آن کلمات انگلیسی که همه بلدیم، Availability و Reliability و غیره روی این بهتر است. هرچند که پشتیبانی بدی دارد و همش می‌خواهد شما را تشنه کند. البته مایکروسافت از تمام این‌ها بدتر است. اولا که ما پشتیبانی نداریم ولی به روزرسانی‌ها منوط به یک فیکس‌لیستی است که هزارتا مشکل را برطرف می‌کند و هزارتا مشکل دیگر را به وجود می‌آورد. هرچند که ما بی‌سوادیم. شما پایداری سیستم فیس‌بوک را نگاه کنید که با چند میلیون مشتری آنلاین تبادل داده نامحدود و باورنکردی می‌کند و شما کندی احساس نمی‌کنید و در بانکداری الکترونیک در پیک کار با ۲۰ هزار نفر که فقط سشن دارند یعنی فقط در سیستم هستند و لزوما کلیک یا فعالیتی در سایت نمی‌کنند شب عید از کار می‌افتد. پس این‌که من می‌گویم بی‌سوادیم باور کنید راست می‌گویم. ما ساده‌ترین چیزها را در این‌جا نمی‌توانیم نگه داریم.

تراکنش را می‌توانید به یک گله تشبیه کنید اگر یک سیستم چابک دارید با تراکنش‌های کوچک زیاد مثل دسته مورچه MySQL از تمام این‌ها بهتر است. اگر تراکنش‌هایی مثل فیل دارید اوراکل و SQL server بهتر است. اگر تراکنش‌های مالی با اهمیت و حجم داشتید DB2 و بعدش oracle جوابگوتر هستند. شاپرک ما الان روی DB2 است. یک ابزاری دارد که ۶ میلیون رکورد مالی را دو دقیقه‌ای لود می‌کند ولی اگر آن را با جاوا انجام دهی دو ساعت لود فایل طول می‌کشد. امنیت و مانیتورینگ خوبی دارد.

ابزارها و مفاهیم جدید مثل NoSQL چطور؟

در این زمینه خیلی متخصص نیستم اسمش را شنیده‌ام اما حتی یک پاراگراف درباره‌اش نخوانده‌ام، خیلی راه داریم تا به آن‌جا برسیم. همین الان در تشخیص این‌که از چه پایگاه‌داده‌ای استفاده کنیم گیر هستیم. بدون این‌که بدانیم الان کجا هستیم و در آینده به کجا می‌خواهیم برویم. این‌که بدون فکر از یک ابزار جدید استفاده کنیم اشتباه است. لزومی ندارد همه به روز باشیم ولی لازم است که چیزی که داریم را خوب نگه داریم و همراه با وندور باشیم.

وضعیت لایسنس نرم‌افزارهایتان چطور است؟

لایسنس را به سختی از طریق شرکتی خریدیم که یک نمایندگی در سوئد داشت. پشتیبانی نمی‌توانیم داشته باشیم. می‌توانیم از طریق سیستمی که آی‌پی‌اش دیده نمی‌شود یک پچ رایگان را دانلود کنیم. مشکل تحریم است. مین‌فریم بیست میلیاردی ممکن است ۴۰ میلیارد تمام شود اما برای بانک ارزش دارد.

مشکل شرکت‌های ایرانی در کارهای مرتبط با پایگاه داده چیست؟

شما باید بیایید از «الف» تا «ی» یک بیزینس را بررسی کنید ببینید کجایش مشکل دارد. تخصص که به اندازه کافی وجود دارد پس چرا سیستم درست کار نمی‌کند؟ اول به خاطر تعامل. دوم گفتن جمله «من مشکلی ندارم». سال ۸۶ من دنبال نیرو می‌گشتم و آقای باغومیان یکی از فعالان سایت تکنوتاکس را معرفی کرد. وقتی آمد و صحبت کردیم من حتی نگذاشتم فرم پر کند. وقتی به او گفتم اگر روزی بخواهی یک برنامه برای غیرلینوکس بنویسی چی. گفت من نمی‌توانم. این تعصب نباید وجود داشته باشد.

چرا بانک‌ها و ای‌تی‌ام‌ها هنوز روی ویندوز مانده‌اند؟

شاید علتش کم کار کردن آن‌هایی است که در بحث متن‌باز کار می‌کنند. این را رسانه‌ای مثل شما باید جا بیندازد. بانک‌ها باید به این سمت بروند. حداقل این‌که آزاد است و می‌توانی شخصی‌سازی‌اش کنی. نه دیگر پول آنتی‌ویروس می‌دهی نه از این بابت نگران هستی. من به عنوان سر دسته این قضیه حاضرم سه نصف شب با یک مدیر جلسه بگذارم و یک پرزنتیشن ارائه بدهم که نه با این دید که خواهش می‌کنم به سمت لینوکس بروید بلکه با این دید که احمق هستید اگر به این سمت نروید. یک نفر باید با قدرت جلو بیفتد و خسته نشود. من سال ۸۳ به همراه آقای تشکری و دوستان دیگر در تیم مهاجرت از ویندوز به لینوکس داده‌پردازی بودم. من آن زمان نماینده اداره کل اینترنت بودم. هدف این بود که تمام ۶۰ منشی، کارهای اداری را با نرم‌افزارهای متن‌باز انجام دهند. قرار شد این آفیس را آموزش دهیم. موفق نشدیم به خاطر دیدمدیریت، چون حمایت نداشتیم. اولین هاستینگ لینوکس داده‌پردازی را من راه‌اندازی کردم و اولین مشتری‌ام هم داده‌پردازی اصفهان بود. مدیریت درست نیست. همان‌طور که بانک در فاکتور توسعه‌اش، دیتاسنتر نمی‌داند چیست و جا کم می‌آورد. از اصول است که وقتی دیتاسنتر طراحی کردید اگر هیچ رشدی هم ندارید ۲۰ درصد بیشتر از چیزی که نیاز داری را در نظر بگیر. ما سال ۸۵ از یک سرور شروع کردیم الان به ۳۵۰ سرور رسیده است. سالی دو نفر افزایش داشتیم ماهی دو سرور. ۵ سال پیش محاسبه کردم دیدم ما ۷ ترابایت نیاز داریم. درخواست ۱۰ ترابایتی دادم. این خریدش دو سال طول کشید. جایش یک ۴۰ ترابایتی دادند. هفته پیش ۳۰ ترابایت کم آمد. طرف صد برابر برای ویندوز و محصولات احمقانه‌اش می‌دهد که تحریم‌ها برداشته شود باید هزار برابر هزینه کند.

شما پایداری سیستم‌فیس‌بوک را نگاه کنید که با چند میلیون مشتری آنلاین تبادل داده نامحدود و باورنکردی می‌کند و شما کندی احساس نمی‌کنید و در بانکداری الکترونیک در پیک کار با ۲۰ هزار نفر که فقط سشن دارند یعنی فقط در سیستم هستند و لزوما کلیک یا فعالیتی در سایت نمی‌کنند شب عید از کار می‌افتد

در بانک‌ها چیزی به نام تراکنش مالی داریم این آیا معادل همان تراکنش دیتابیس است؟

نه شما شاید در ۲ دیتابیس مختلف بابت یک کار کاربر سه رکورد ثبت کنید. برای مسائلی مانند وب‌کاوی، کشف تقلب، داده‌کاوی لاگ اتفاقات را هم ثبت می‌کنند. یعنی از چه مرورگری با چه آی‌پی چه کاری انجام داده است.

یعنی وقتی پیام می‌آید که تراکنش با موفقیت انجام شد این تراکنش مربوط به دیتابیس است یا چیزی غیر از دیتابیس هم این وسط وجود دارد؟

ممکن است به خاطر این‌که کاربر معطل ثبت در دیتابیس نشود یک روال آفلاینی دربیاوریم که با یک تأخیر کوچکی در یک فایل ذخیره شود و از فایل به دیتابیس برود. در نهایت مرجع دیتابیس است. ولی لاگ مشتری‌ها در یک فایل ثبت می‌شود و با یک اپلیکیشن دیگری هر چند دقیقه یکبار به‌روزرسانی می‌شود.

وقتی که یک تراکنش ناموفقی داریم رول بک چرا این‌قدر طول می‌کشد؟

این سؤال اصلا به جا نیست جایی که تراکنش مالی‌اش ۲۴ ساعت طول می‌کشد می‌خواهید رول‌بک‌اش ۴۸ ساعت طول نکشد؟ شاپرک در روز در ۷ سیکل پول را واریز می‌کند.

حجم و تعداد تراکنش‌های شما چقدر است؟

شاید در طول روز بانک ملت ۲۰ میلیون تراکنش مالی داشته باشد که از طریق شاپرک و پذیرنده‌ها انجام می‌شود.

تخصیصی

در بدترین شرایط هم می‌شود از نو ساخت

نحوه احیای سرور

فقـط بـرای تفریح، تصمیم گرفتـم که یک سرور لینوکس راه بندازم و دستور rm -rf / را با کاربر ریشه (root) اجرا کنم و ببینم چه چیزهایی باقی می‌ماند. فهمیدم کـه rm به خرابکارهایی مثل مـن فکر کـرده و بـرای انجـام همچیـن کاری باید no-preseve-root -- را اعمال کنیم.

```
# rm -rf --no-preserve-root /
```

بعـد از اجرای این دسـتور مسـخره، ابزارهای خوبی مثل

```
/bin/ls
/bin/cat
/bin/chmod
/usr/bin/file
```

همگـی از بین میرود! البته همچنـان ارتباط جاری SSH و بش و در نتیجه تمام ابزارهای توکار بش مثل echo را در اختیار داریم.

```
root@rmrf:/# ls
-bash: /bin/ls: No such file or directory
```

دیگر ls نداریم، ولی echo و fileglobs ها هنوز هستند. با این‌ها چیکار می‌تونیم بکنیم؟

```
root@rmrf:/# echo *
dev proc run sys
# echo /dev/pts/*
/dev/pts/0 /dev/pts/3 /dev/pts/ptmx
```

آهـان! مـا dev/ و run/ و proc/ و sys/ را داریـم. و حالا که ls داریم، می‌تونیم کاری کنیم که خواناتر باشه.

```
root@rmrf:/# for file in /dev/pts/*; do echo $file;
done
/dev/pts/0
/dev/pts/3
/dev/pts/ptmx
```

و البتـه از آنجایـی کـه printf هنوز هسـت و در بش می‌شود تابع تعریف کرد، ما ابزاری مثلی ls می‌سازیم اگرچه خیلی محدودتر است.

```
root@rmrf:/# ls() { printf '%s\n' ${1:+${1%/}/}*; }
-bash: syntax error near unexpected token `('
```

چی؟ دستور کاملا درسته و مشکلی نداره. ببینیم ls هش یا الیاس شده؟

```
root@rmrf:/# type ls
ls is aliased to `ls --color=auto'
```

اوه دستور در واقع این می‌شود

```
ls--color=auto () { printf '%s\n' ${1:+${1%/}/}*; }
```

خوب می‌توانیم آن را از الیاس خارج کنیم.

```
root@rmrf:/# unalias ls
root@rmrf:/# ls() { printf '%s\n' ${1:+${1%/}/}*; }
root@rmrf:/# ls
/dev
/proc
/run
/sys
root@rmrf:/# ls /dev
/dev/pts
```

خوب کارمون را تا این جا می‌توانیم ذخیره کنیم.

```
root@rmrf:/# echo 'ls() { printf '%s\n' ${1:+${1%/}/}*;
```

```
}' >> utils.sh
root@rmrf:/# source utils.sh
```

و cat چطـور؟ read بـه صـورت تـوکار وجـود داره و می‌توانیـم بـا pipe هـا و redirection یـه cat سـاده و ابتدایی بسازیم

```
root@rmrf:/# (while read line; do echo "$line"; done)
< utils.sh
ls() { printf '%s\n' ${1:+${1%/}/}*; }
```

بـا ایـن امکانـات و این حقیقت که مـا می‌توانیم هر بایتـی کـه بخواهیم را با echo بنویسـیم، می‌توانیم باینری‌های curl یا wget را دوباره مسـتقیم بسـازیم. اولیـن انتخـاب مـن echoed by others، گرفتـن busybox خواهـد بـود. busybox یـه چاقوی ارتشـی سویسی (آچار فرانسه)ه لینوکس‌های امبدد است با ابزارهـای تـوکاری مثـل wget و dd و tar و کلی ابزار دیگر.

الان یک مشکلی هست.

حتـی اگر مـا همه بایت‌هایی که نیاز داریـم را echo کنیم که تمام باینری‌ها را بسـازیم، آن فایل‌ها قابل اجرا نیستند. هیچ راهی برای اجرای busybox نداریم. سـاده‌ترین راه‌حل بـرای این، پیدا کـردن یک فایل قابل اجراست که آن را با echo بازنویسی کنیم. البته در این مرحله ما usr/ و bin/ را نابود کردیم، در نتیجه برای این کار، کمی ترفند نیاز است.

می‌توانیـم از globها و منطق بش اسـتفاده کنیم تا فایل‌هایی که بیت قابل اجرا بودن دارند را پیدا کنیم، و البتـه مطمئن شـویم کـه دایرکتوری‌هـا را نادیده می‌گیریم.

```
executable () { if [[ ( ! -d $1 ) && -x $1 ]] ; then echo
"$1"; fi }
```

و پیدا کردن قابل اجراها!

```
root@rmrf:/# for file in /*; do executable $file; done
root@rmrf:/# for file in /*/*; do executable $file;
done
root@rmrf:/# for file in /*/*/*; do executable $file;
done
/proc/1107/exe
/proc/1136/exe
/proc/1149/exe
/proc/1179/exe
/proc/1215/exe
/proc/1217/exe
/proc/1220/exe
/proc/1221/exe
/proc/1223/exe
/proc/1248/exe
/proc/1277/exe
/proc/1468/exe
/proc/1478/exe
/proc/1625/exe
/proc/1644/exe
/proc/1/exe
/proc/374/exe
/proc/378/exe
/proc/471/exe
/proc/616/exe
/proc/657/exe
/proc/self/exe
```

عالی! صبر کنید یک لحظه، همه این‌ها اشاره دارند به فایل‌های قابل اجرایی که دیگر وجود ندارند.
خوب پس executable() رو تغییر می‌دهیم تا لینک‌ها رو نادیده بگیره.

```
root@rmrf:/# executable () { if [[ ( ! -d $1 ) && ( ! -h
$1 ) && -x $1 ]] ; then echo "$1"; fi }
root@rmrf:/# for file in /*/*/*; do executable $file;
done
root@rmrf:/# for file in /*/*/*/*; do executable $file;
done
root@rmrf:/# for file in /*/*/*/*/*; do executable
$file; done
root@rmrf:/# for file in /*/*/*/*/*/*; do executable
$file; done
```

خوب الان، این خبر بدی است.
از سیستم سالم با معماری box یکسان:

```
$ mkdir $(xxd -p -l 16 /dev/urandom)
$ cd $_
$ apt-get download busybox-static
$ dpkg -x *.deb .
$ alias encode='{ tr -d \\n | sed "s#\\(..\\)#\\\\x\\1#g";
echo; }'
$ alias upload='{ xxd -p | encode | nc -q0 -lp 5050; }'
$ upload < bin/busybox
```

و در سیستم rmrf شده:

```
# cd /
# alias decode='while read -ru9 line; do printf
"$line"; done'
# alias download='( exec 9<>/dev/tcp/{IP OF NON
HOSED BOX}/5050; decode )'
# download > busybox
```

الان می‌توانیم یک Object مشترک بسازیم که روی busybox مجوزها را تغییر می‌دهد:

```
$ cat > setx.c <<EOF
extern int chmod(const char *pathname, unsigned
int mode);

int entry(void) {

    return !! chmod("busybox", 0700);
}
char *desc[] = {0};

struct quick_hack {

    char *name; int (*fn)(void); int on;
    char **long_doc, *short_doc, *other;

} setx_struct = { "setx", entry, 1, desc, "chmod 0700
busybox", 0 };
EOF
$ gcc -Wall -Wextra -pedantic -nostdlib -Os -fpic
-shared setx.c -o setx
$ upload < setx
```

وقـت فعـال کردن setx به عنوان توکار اسـت و قابل اجرا کردن busybox:

```
# ( download > setx; enable -f ./setx setx; setx; )
# /busybox mkdir .bin
# /busybox  --install -s .bin
# PATH=/.bin
```

ایمکس جادوی گنو

نویسنده:
سمیر رحمانی

اگر کاربر گنو لینوکس هستید، حتما نام گنو ایمکس (GNU/Emacs) را شنیده‌اید. نرم‌افزاری که به واسطه امکانات و قابلیت‌های بی‌شمار و فوق العاده‌اش طرفداران زیادی را در دنیای نرم‌افزارهای آزاد به خود اختصاص داده است. ایمکس ۳۹ ساله، به نوعی قدیمی‌ترین نرم‌افزار آزاد محسوب می‌شود که پس از گذشت این همه سال هنوز با حجم بالایی از مشارکت‌ها برای توسعه آن روبه‌روست. در این مقاله بر آن شدم که نگاهی عمومی بر برخی ازویژگی‌های این ویرایشگر (editor) دوست داشتنی بیاندازیم.

پشتیبانی از همه زبان‌ها

با پشتیبانی از Unicode ایمکس تقریبا همه زبان‌های دنیا از جمله فارسی را به راحتی پشتیبانی می‌کند. علاوه بر این وجود افزونه‌هایی برای اصلاح دستور زبان و بررسی صحت نوشتار کلمه‌ها، محیط جذابی را برای نوشتن متون مختلف در اختیار کاربر قرار می‌دهد.

توسعه آسان و سریع

ایمکس با استفاده از زبان برنامه نویسی C پیاده‌سازی شده است. شاید با خود بگویید، چگونه ممکن است برنامه‌ای که با زبان C نوشته شده است، روند توسعه سریع و آسانی داشته باشد؟، پاسخ این سوال بسیار ساده است. ایمکس از زبانی بر پایه lisp برای گسترش و توسعه عملکردهایش، با نام ایمکس - لیسپ یا به اختصار ای-لیسپ (Elisp) بهره می‌برد. این زبان برخلاف ظاهر ترسناکش بسیار ساده و دلچسب بوده و طی مدت کوتاهی می‌توان آن را یاد گرفت.

ای-لیسپ زبانی است که منحصرا برای توسعه ویرایشگر ایمکس ساخته شده است وعلاوه بر داشتن ویژگی‌های یک زبان برنامه‌نویسی کامل، توابع بسیار زیادی را برای کاربردهای مختلفی نظیر پردازش متن، ویرایش رابط کاربری، ارتباط با نرم‌افزارهای خارجی، ارتباط با شبکه‌های خارجی و... را مهیا می‌سازد. با توجه به این موضوع ساختن یک افزونه جدید برای ایمکس بسیار ساده و سریع است. به طوری که در هنگام استفاده از ایمکس می‌توانید کد مورد نظر خود را به زبان ای-لیسپ نوشته وهمانجا اجرا کنید و در همان لحظه تاثیر آن را بر روی ایمکس خود مشاهده کنید. شاید برای شما هم بارها اتفاق افتاده باشد که در متنی، بخواهید با استفاده از الگوی مشخصی تغییراتی ایجاد کنید. با استفاده از ای-لیسپ به راحتی می‌توانید در همان متن با چند خط کد ای-لیسپ به هدف خود برسید.

افزونه‌های متعدد

با توجه به راحتی روند توسعه افزونه در ایمکس، افراد بسیاری با نیازها مختلف اقدام به ساخت افزونه‌های گوناگونی برای این ویرایشگر نموده‌اند. تعداد این افزونه‌ها به قدری زیاد است که حتی یک کاربر کهنه کار و حرفه‌ای ایمکس را نیز شگفت زده می‌کند. به طوری که هر روز می‌توان کارایی و افزونه جدیدی را بر روی ویرایشگر خود نصب کنید. در نسخه‌های اخیر این ویرایشگر، نرم‌افزار مدیریت بسته‌های نرم‌افزاری با نام (Package.el) به ایمکس اضافه شده که به کاربر اجازه می‌دهد بدون هیچ دردسری اقدام به مدیریت افزونه‌های خویش نماید. شاید ایمکس اولین ویرایشگر متنی باشد که چنین قابلیتی را عرضه نموده است. گرچه این ویژگی در ویرایشگرهای دیگر هم دیده می‌شود.

کاربردهای گوناگون

بر خلاف ویرایشگرهای دیگر که تنها برای ویرایش متن ساخته شده‌اند، ایمکس داستان متفاوتی را با خود به یدک می‌کشد. جامعه کاربری بزرگ این نرم‌افزار سبب شده تا افراد مختلفی با سلیقه‌های متفاوت کنار یکدیگر جمع شوند و ایمکس را به گونه‌ای توسعه دهند که پاسخی باشد برای نیازهایشان.
از این رو از ایمکس می‌توان علاوه بر ویرایش متن برای کاربردهای دیگری نیز بهره برد. کاربردهایی نظیر:

- نمایش تصاویر
- مدیریت پروژه
- پخش‌کننده موزیک و ویدئو
- نمایش مستندات مختلف از جمله PDF
- مرورگر وب
- چت کلاینت
- SPREADSHEET
- ایجاد اسلاید
- رسم نمودار
- ارسال و دریافت ایمیل
- .....

مطمئنا کاربردهای خارق‌العاده دیگری نیز از این فهرست جامانده‌اند.

محیط توسعه

اگر برنامه‌نویس هستید و از محیط‌های توسعه بسیار سنگین و کند خسته شده‌اید و همچنین به امکاناتی بیش از یک ویرایشگر متن نیاز دارید. ایمکس چاره مشکل شماست. ایمکس با حجمی بسیار ناچیز در مقابل محیط‌های توسعه دیگر از سرعت بسیار بالایی بر خوردار است. به صورتی که شگفت زده خواهید شد. محیط توسعه‌ای را در نظر بگیرید که از زبان‌های بی‌شماری پشتیبانی می‌کند. سورس کد شماره در زمان نوشتن کد همواره برای یافتن خطاهای دستوری و یا مشکلات مربوط به ساختار کد و نحوه نوشتن کد بررسی می‌کنند و خطاها را به شما گزارش داده و روش‌های مناسب‌تر را به شما پیشنهاد می‌دهد. با زدن یک دکمه مستندات تابع با کلمه کلیدی مورد نظر را به شما نشان می‌دهد. برای ادامه کد، بر اساس کد شما پیشنهادات هوشمندانه را ارائه داده و کد را کامل کند. کد را کامپایل و اجرا کرده و دیباگر مناسبی را برایتان فراهم می‌کند. از ابزارهای مدیریت پروژه شما، آخرین وظایفتان را دریافت کرده و به شما یادآوری می‌کند. تمام این قابلیت‌ها را به همراه بازه بسیار وسیعی از توانایی‌های دیگر را می‌توانید در کمترین زمان توسط ایمکس داشته باشید.

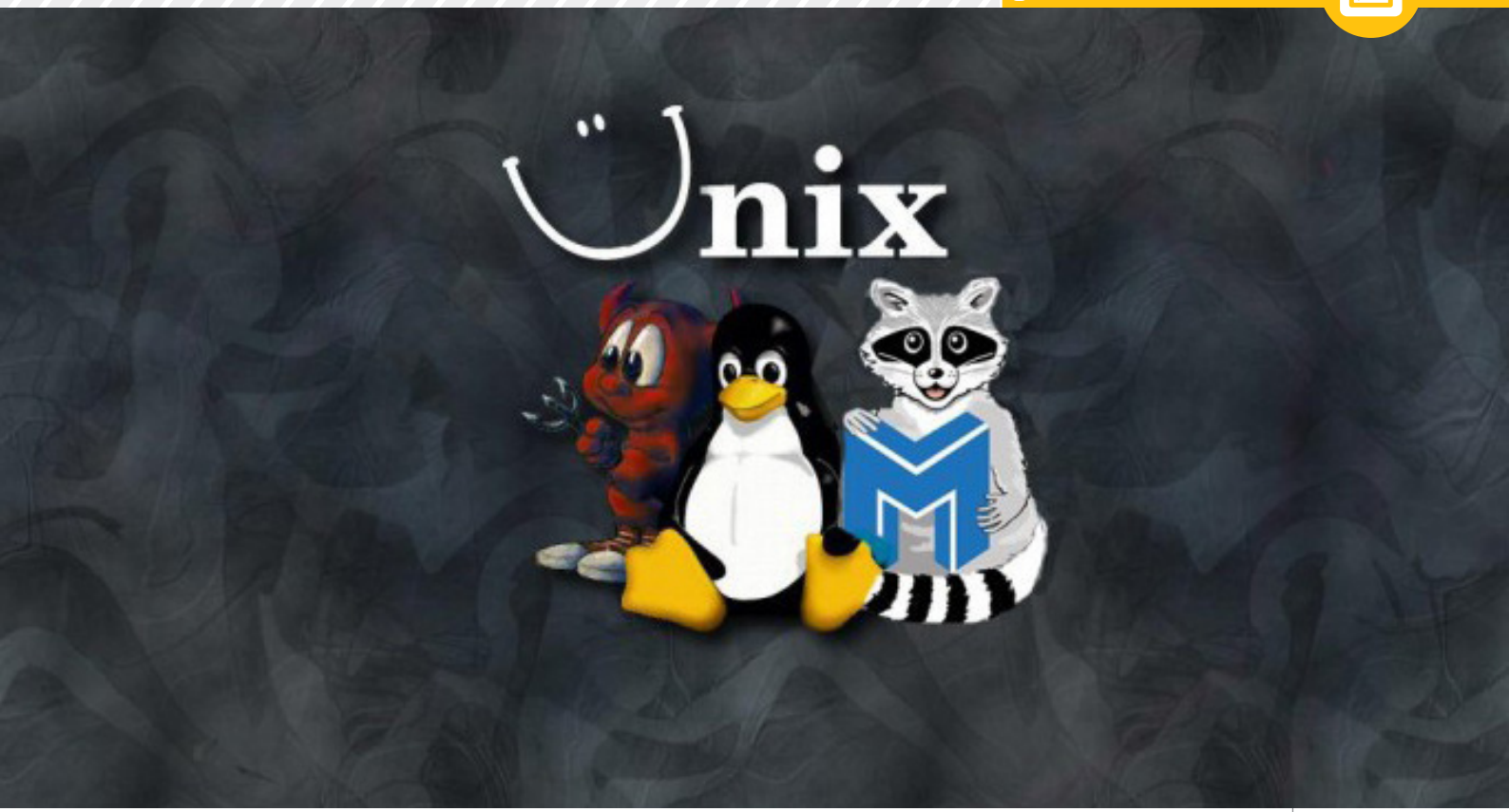
جامعه کاربری گسترده

به واسطه قابلیت‌های بسیار زیاد ایمکس که نگاه مختصری بر مواردی از آن‌ها داشتیم، ایمکس کاربران زیادی در دنیا به خود اختصاص داده است. کاربرانی با سطوح مختلف دانش و نیازهای متفاوت. ایمکسی که امروز ما از آن بهره می‌بریم حاصل تلاش هزاران نفر است. چنین جامعه گسترده و فعالی ویژگی درخشانی برای یک نرم‌افزار به شمار می‌آید.

نقطه شروع

ایمکس دنیایی است متفاوت از آن‌چه تاکنون تجربه کرده‌اید. از این رو، کاربرانی که تازه وارد دنیای ایمکس می‌شوند، غالبا نمی‌توانند با بعضی از مفاهیم این نرم‌افزار ارتباط برقرار کرده و از تجربه خود در این دنیا لذت برند. ایمکس به صورت پیش‌فرض تنظیمات خاصی ندارد و بسیار ساده و ابتدایی به نظر می‌رسد و همان طور که اشاره شد حجم بسیار زیادی از افزونه‌ها می‌تواند برای یک کاربر تازه کار گیج کننده باشد.
اما نگران نباشید. کاربران حرفه‌ای تر در جامعه ایمکس برای این مشکل چاره اندیشیده‌اند و توزیع‌های مختلفی را برای استفاده افراد تازه کار منتشر نموده‌اند. توزیع‌هایی که حتی کاربران حرفه‌ای را نیز مجذوب خود کرده است.
این توزیع‌ها تنظیمات متعددی را بر روی ایمکس انجام می‌دهند و افزونه‌های پرکاربرد را برای شما نصب و پیکربندی می‌کنند و می‌توانند نقطه مناسبی برای شروع باشند. در ادامه به تعدادی از این توزیع‌ها اشاره می‌شود:
Emacs starter kit – توزیع بسیار ساده و ابتدایی که کاربران تازه کار را هدف قرار داده است.
Prelude – این توزیع یکی از پرطرفدارترین توزیع‌های ایمکس است که همه کاربران را هدف قرار می‌دهد. استفاده از این توزیع را به شما پیشنهاد می‌کنم.
Spacemacs – این توزیع با هدف پوشش کاربرانی که از ویرایشگر vim استفاده می‌کنند ساخته شده و طرفداران زیادی نیز دارد.
اگر از vim استفاده می‌کنید و دوست دارید ایمکس را نیز تجربه کنید این توزیع را حتما امتحان کنید.
FG42 – توزیعی جوان که برای کاربران مختلف توسعه داده می‌شود. این توزیع روی کرد متفاوتی نسبت به بقیه دارد و توسعه آن نیز آسان‌تر است. استفاده از این توزیع رو به شما پیشنهاد می‌کنم.
در نظر داشته باشید استفاده از این توزیع‌ها به تنهایی برای شروع کافی نیست و شما باید مقداری در رابطه با مفاهیم ابتدایی ایمکس مطالعه نمایید. وب سایت emacswiki.org مکان مناسبی برای شروع مطالعه است.
در آخر باید اضافه کرد که ایمکس دنیایی فراتر از آن چیزی است که ممکن است تصور کنید. دنیای بسیار جذاب و دوست داشتنی که بعد از مدتی دیگر نمی‌توانید از آن دل بکنید. به شما پیشنهاد می‌کنم اگر تا الان از ایمکس استفاده نمی‌کردید، به هر نحوی که شده مدتی از این ویرایشگر قدرتمند و دوست داشتنی استفاده کنید تا دنیای متفاوتی را تجربه کنید. ■

ایمکسی که امروز ما از آن بهره می‌بریم حاصل تلاش هزاران نفر است. چنین جامعه گسترده و فعالی ویژگی درخشانی برای یک نرم‌افزار به شمار می‌آید.



نگاهی اجمالی بر معماری سیستم‌عامل‌ها

رایانه‌ها را می‌توان یکی از مهمترین اختراعات و ابداعات بشری دانست. آنان در سطوح گسترده‌ای در زندگی روزانه ما نفوذ یافته‌اند و تمامی امور روزانه و حیاتی بشر را در چنته دارند. رایانه‌های امروزی دیگر مانند سابق مختص به رایانه‌های شخصی قدیمی و بزرگ که بر روی میز قرار می‌گرفتند و کاربردهای خاصی داشتند، نبوده و در گستره گسترده‌ای از کاربردهای مختلف در حال ارائه خدمات هستند. به عنوان مثال زمانی که از گوشی‌هوشمند خود استفاده می‌کنید؛ نیز از یک رایانه استفاده می‌کنید که از معماری آرم بهره می‌برد. رایانه‌ها در هر شکل و ساختاری از دو بخش تشکیل شده‌اند؛ سخت‌افزار و نرم‌افزار، که کم‌وبیش با این موضوع آشنا هستید. بااین‌حال بخش نرم‌افزاری نیز خود به چند قسمت دیگر قابل تقسیم است که در زیر به آنان اشاره شده است.

به طور کلی نرم‌افزارهای رایانه را می‌توان به دو دسته عمده تقسیم کرد

نرم‌افزارهای کاربردی که کارهای مورد نیاز کاربران را انجام‌می‌دهند؛

نرم‌افزارهای سیستمی که وظیفه کنترل و مدیریت خود رایانه را بر عهده دارند؛

سیستم‌عامل «Operating System»، اساسی‌ترین برنامه سیستمی است که مدیریت کلیه منابع سیستم را به عهده گرفته و زمینه‌ای فراهم می‌سازد که برنامه‌های کاربردی بتوانند بر روی آن نوشته شوند.

دراین نوشتار قصد دارم به بررسی اجمالی معماری یکپارچه لینوکس و سپس ریزهسته و مینیکس به‌علاوه نحوه کار آن و سپس توضیحات مفصل در این‌باره بپردازم که در برخی موارد در انتهای مقاله ممکن است کاملا تخصصی باشد. بنابراین مخاطب اصلی مقالات فوق را افرادی تشکیل خواند داد که حداقل دو واحد درس سیستم‌عامل را پاس کرده باشند.

ساختار و معماری سیستم‌عامل‌ها

ساختارهای مختلفی در این بخش وجود دارد که ما به بررسی مختصر ساختار مربوط به لینوکس که هسته سیستم‌عامل گنو/لینوکس است و سیستم‌عامل مینیکس که نمونه‌ای خوب برای ساختار ریزهسته است؛ خواهیم پرداخت. ساختارهای عمده موجود عبارتند از :

- ساختار یکپارچه
- ساختار پیوندی*
- ساختار مشتری خدمتگزار «ریزهسته»

سیستم‌های یکپارچه

می‌توان این روش متداول در گذشته را «درهم‌ریختگی بزرگ» نامید. ساختار آن در بی‌ساختاری نهفته است. سیستم‌عامل به صورت مجموعه‌ای از روندها نوشته شده که هر یک از آن‌ها می‌تواند به فراخور نیاز، دیگری را صدا بزند وقتی از این تکنیک استفاده شود هر فرآیند در سیستم، یک واسط دارد که شامل مولفه‌ها و نتایج خروجی است که به شکلی ساده تعریف شده است. هر یک از آنان قادر است تا دیگری را فراخوانی کند تا محاسبات لازم را برایش انجام دهد.

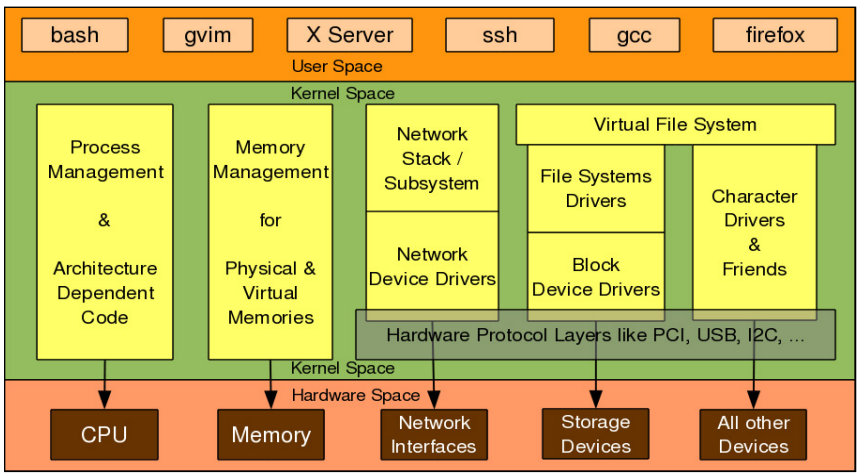
وقتی از این تکنیک استفاده شود برای ساخت برنامه شیء «آبجکت» واقعی سیستم‌عامل، ابتدا باید روندهای جداگانه و یا فایل‌های شامل روندها را ترجمه نماییم و سپس با استفاده از پیونددهنده «Linker» سیستم، همه آن‌ها را به هم پیوندزده و در یک فایل آبجکت یکتا قرار دهیم. برای مخفی کردن اطلاعات، امکانات خاصی وجود ندارد و هر رویه برای دیگر رویه‌ها کاملا قابل مشاهده است.

به هر حال حتی در سیستم‌های یکپارچه، امکان داشتن حداقل یک ساختار کوچک وجود دارد. برای تقاضای سرویس‌های «فراخوان سیستمی» فراهم شده توسط سیستم‌عامل، مولفه‌های مرتبط را در مکان‌های دقیقا تعریف شده مانند ثبات‌ها یا پشته قرار می‌دهیم و سپس یک ساختار تله «trap instruction» مخصوص را اجرا می‌کنیم. که «Supervisor call» و «Kernel call» نام دارند. این دستورالعمل، ماشین را از مد کاربر به مد هسته تغییر حالت داده و کنترل را در اختیار سیستم‌عامل قرار می‌دهد.

ساختار پیوندی:

ساختار پیوندی «Hybrid» ساختاری برای هسته سیستم‌عامل است که هم ویژگی‌های ریزهسته و هم ویژگی‌های هسته‌های یکپارچه را داراست.

مدل‌های سنتی مورد استفاده در هسته سیستم‌عامل‌ها، مدل‌های ریزهسته و هسته یکپارچه یا همان یکپارچه هستند. برخی افراد است ساختار را همان ساختار یکپارچه می‌دانند؛ ایده اصلی پشت این ساختار این است که هسته از مزایای ریزهسته برخوردار باشد، اما به دلیل مشکلات اجرایی ریزهسته‌ها در عمل به صورت هسته‌ای یکپارچه پیاده‌سازی شود. برخلاف ریزهسته‌ها، در یک هسته پیوندی، همه (یا تقریباً همه) سرویس‌های سیستم‌عامل در بخش فضای هسته قرار دارند (مشابه هسته یکپارچه). برخلاف هسته‌های یکپارچه، نمی‌توان در زمان اجرا بودن سیستم، ماژول‌هایی را به هسته‌های پیوندی حذف یا اضافه کرد.



ارائه توضیحات بیشتر در مورد ساختار پیوندی از موضوع این بحث خارج است و چون ساختار هسته لینوکس از ساختار یکپارچه بهره می‌برد، مجبور به ارایه توضیحاتی بیشتر در مورد ساختار یکپارچه شدم. برای خواندن توضیحات بیشتر درباره ساختار پیوندی به کتاب سیستم‌های عامل نوشته تنن‌باوم «Andrew Tanenbaum» مراجعه کنید.

ساختار ریزهسته «میکروکرنل»

مینیکس ویرایش سوم، سیستم‌عاملی متن‌باز است که بسیار منعطف، قابل اطمینان و امن می‌باشد. این سیستم‌عامل از نظر پایه «Base» چیزی شبیه به نسخه‌های قبلی خود اما در بسیاری از جهات کاملا متفاوت است. مینیکس ویرایش اول و دوم بیشتر برای ابزار آموزشی در نظر گرفته شده بودند , اما هدف مینیکس ویرایش سوم به سمت یک سیستم‌عامل معمولی بودن نشانه رفته شده است.

سیستم‌عامل جدید بی‌نهایت کوچک است. قسمت‌هایی که در مد کاربر اجرا می‌شوند به ماژول‌های کوچکتر تقسیم و به خوبی از هم جدا شده‌اند، برای مثال راه‌انداز هر وسیله در مد«حالت»-کاربر مجزا پردازش می‌شود، بنابراین وجود مشکلات در یک راه‌انداز کل سیستم‌عامل را تحت‌الشعاع قرار نمی‌دهد. در حقیقت در اکثر مواقع زمانی که یک راه‌انداز با مشکل مواجه می‌شود به صورت خودکار بدون نیاز به مداخله کاربر و یا راه‌اندازی مجدد سیستم، جایگزین می‌شود. حتی این روند تأثیری بر روی برنامه‌های در حال اجرا هم نخواهد گذاشت. این ویژگی‌ها در هسته‌ای با این حجم کوچک کد منبع، یک سیستم قابل اعتماد را برای ما ایجاد خواهد کرد.

اکنون می‌توانیم به منظور تکمیل مطالعات خود در زمینه مدیریت فرآیند، ارتباط بین فرآیندها و زمان‌بندی، به بررسی چگونگی استفاده از آن‌ها در مینیکس ویرایش سوم بپردازیم. بر خلاف یونیکس که هسته آن یکپارچه است و به ماژول‌های مختلف تجزیه نشده، مینیکس ۳ مجموعه‌ای از فرآیندهاست که با یکدیگر و با فرآیندهای کاربر، از طریق یکی از روش‌های اولیه ارتباط به نام تبادل پیغام، ارتباط برقرار می‌کنند. این طراحی یک ساختار پیمانه‌ای «ماژولار» با قابلیت انعطاف بیشتر را به وجود می‌آورد که آن را ساده‌تر می‌سازد. به عنوان مثال می‌توان کل سیستم‌فایل را بدون نیاز به ترجمه «کامپایل» مجدد هسته با یک سیستم کاملاً جدید دوباره نویسی و جایگزین کرد؛ به عبارت دیگر اگر قسمتی از سیستم‌عامل مانند سیستم‌فایل به‌روز شود، می‌توان از امکانات و بهبودهای جدید آن بدون نیاز به دستکاری هسته استفاده کرد.

همچنین افزودن خدمت‌گزارهای جدید مستلزم ترجمه مجدد کل سیستم نیست. می‌توان سیستم مدیر فرآیند و سیستم‌فایل را با خدمت‌گزار شبکه و سایر خدمت‌گزارها تکمیل نمود و این امر به وسیله اضافه نمودن خدمت‌گزارهای اضافی مورد نیاز صورت می‌گیرد. اگرچه معمولاً گرداننده‌های دستگاه در هنگام بارگزاری سیستم، راه‌اندازی می‌شوند اما می‌توانند بعداً نیز

مدل‌های سنتی مورد استفاده در هسته سیستم‌عامل‌ها، مدل‌های ریزهسته و هسته یکپارچه یا همان یکپارچه‌هستند

به سیستم افزوده شوند. هم گردانندههای دستگاه و هم خدمتگزارها ترجمه میشوند و به عنوان فایلهای اجرایی معمولی بر روی دیسک ذخیره میشوند، اما وقتی که به طور مناسب نصب میشوند از دسترسیهای ممتاز مخصوص خود برخوردار خواهند بود. یک برنامه کاربر به نام «سرویس»، واسط مربوط خدمتگزار تناسخ، که مدیریت این امور محوله را فراهم میکند. اگر چه گردانندهها و خدمتگزارها فرآیندهای مستقلی هستند اما معمولاً با فرآیندهای کاربر، این تفاوت را دارند که تقریبا در تمام مواقع در زمان فعال بودن سیستم، خاتمه نمییابند.

ما اغلب به گردانندهها و خدمتگزارهای درون لایههای ۲ و ۳، تحت عنوان فرآیندهای سیستم اشاره میکنیم. فرآیندهای سیستم بخشی از سیستمعامل به شمار میروند. آنها به هیچ کاربری تعلق ندارند و بسیاری از آنها «البته نه تمامی آنها» قبل از برقراری ارتباط از سوی اولین کاربر، فعال میشوند. تفاوت دیگر فرآیندهای سیستم و فرآیندهای کاربر این است که فرآیندهای سیستم از اولویت اجرایی بالاتری نسبت به فرآیندهای کاربر برخوردار هستند. در حقیقت، معمولاً گردانندهها، اولویت اجرایی بالاتری نسبت به خدمتگزارها دارند، اما این امر به صورت خودکار تحقق نمییابد. اولویت اجرایی در مینیکس ۳ به صورت موردی تعیین میشود؛ ممکن است یک گرداننده که خدمات یک دستگاه کُند را بر عهده دارد، اولویتی پایینتر نسبت به خدمتگزاری که باید به سرعت پاسخ دهد داشته باشد.

این مزیت در هورد «HURD» نیز وجود دارد، اما هسته لینوکس از ساختار مشابه یونیکس و بیاسدی یعنی ساختار یکپارچه استفاده میکند.

ساختار داخلی مینیکس سه

اجازه دهید برای مطالعه مینیکس سه از بالا به آن نگاه کنیم. مینیکس سه از چهار لایه تشکیل شده است که هر لایه وظیفه کاملاً تعریف شدهای دارد. هسته در لایه پایین وظیفه کنترل زمانبندی فرآیندها، مدیریت جابجایی بین حالات آماده، در حال اجرا و بلوکه را بر عهده دارد. همچنین هسته پیغامهای بین فرآیندها را مدیریت میکند؛ مدیریت پیغامها مستلزم بررسی گیرنده میباشد. بخشی از هسته دسترسی به درگاههای ورودی/خروجی و وقفهها را پشتیبانی میکند که در پردازندههای پیشرفته مستلزم استفاده از دستورالعملهای ممتاز مد هسته است و برای فرآیندهای معمولی خارج از دسترس میباشد.

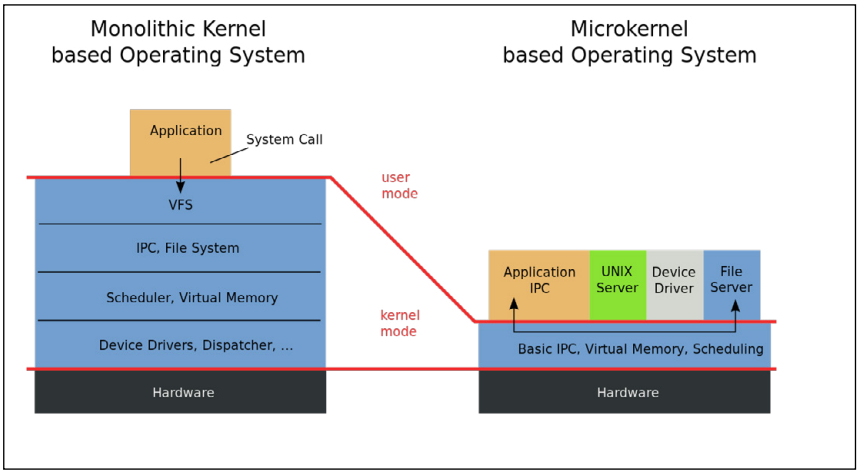
علاوه بر خود هسته، این لایه شامل دو ماژول دیگر است که عملکردی مشابه با گردانندههای دستگاه دارند. وظیفه ساعت یک گرداننده ابزار ورودی /خروجی است به این معنی که با سختافزاری که سیگنالهای زمان را تولید میکند، در تعامل است، اما گردانندههای دیسک یا خطوط ارتباطی، در دسترس کاربر نیست و تنها با هسته در ارتباط است.

یکی از وظایف اصلی لایه اول، فراهم نمودن مجموعهای از فراخوانهای ممتاز هسته برای گردانندهها و خدمتگزارهای قرار گرفته در بالای آن میباشد.این مجموعه شامل خواندن و نوشتن بر روی درگاه ورودی / خروجی، کپی اطلاعات بین فضاهای آدرس و نظیر اینها است. پیادهسازی این فراخوانها به وسیله وظیفه سیستم انجام میشود. اگر چه وظیفه سیستم و وظیفه ساعت در فضای آدرس هسته ترجمه میشوند، اما آنها به عنوان فرآیندهای جداگانه زمانبندی شده و پشتههای فراخوانی خودشان را دارند.

بیشتر هسته و کل وظایف ساعت و سیستم به زبان سی نوشته میشوند. البته بخش کوچکی از هسته به زبان اسمبلی نوشته شده است. این بخشهای زبان اسمبلی به اداره وقفهها، سازوکارهای سطح پایین مدیریت تعویض متن بین فرآیندها «ذخیره و بازیابی ثباتها و امثال آن» و بخشهای سطح پایین کار با ثباتهای سختافزاری MMU میپردازند. کد اسمبلی، اداره بخشهایی از هسته را بر عهده دارد که مستقیماً مرتبط با سختافزار و در سطح بسیار پائین هستند و قادر نیستند به زبان سی بیان شوند.

سه لایه روی هسته میتوانند به صورت یک لایه منفرد در نظر گرفته شوند زیرا هسته اساساً به یک صورت با آنها رفتار میکند. هر سه لایه به دستورالعمل مد کاربر محدود میشوند و همگی توسط هسته زمانبندی میشوند. هیچ یک از آنها به طور مستقیم به درگاه ورودی / خروجی دسترسی ندارند. همچنین، هیچ کدام از آنها نمیتوانند به فضای حافظه خارج از قطعه تخصیص یافته به خود، دسترسی پیدا کنند.

با این حال، فرآیندها به صورت بالقوه امتیازات خاصی دارند «مانند قابلیت اجرای فراخوانهای هسته». این تفاوت واقعی بین فرآیندهای لایه، دوم، سوم و چهارم است. فرآیندهای لایه دوم، از حداکثر امتیاز برخوردارند. در لایه ۳، امتیازات کمتری وجود دارد و در لایه ۴ هیچ امتیاز ویژه ای وجود ندارد. به عنوان مثال فرآیندهای لایه ۲ که گردانندههای دستگاه نام دارند، اجازه دارند که از وظیفه سیستم درخواست خواندن و نوشتن دادهها بر روی درگاه ورودی / خروجی مربوط به خود کنند. برای هر نوع دستگاه از قبیل دیسکها، چاپگرها، ترمینالها و واسطهای شبکه، یک



.این مجموعه شامل خواندن و نوشتن بر روی درگاه ورودی / خروجی، کپی اطلاعات بین فضاهای آدرس و نظیر اینها است. پیادهسازی این فراخوانها به وسیله وظیفه سیستم انجام میشود

گرداننده مخصوص لازم است. بدیهی است اگر دستگاه دیگری اضافه شود به گرداننده خاص خود نیاز دارد. هم‌چنین گردانندههای دستگاه می‌توانند سایر فراخوان‌های هسته مانند درخواست کپی داده‌های وارد شده جدید به فضای آدرس فرآیند متفاوت را اجرا کنند.

لایه سوم شامل خدمت‌گزارها می‌شود. خدمت‌گزارها فرآیندهایی هستند که ارایه خدمات مفید به فرآیندهای کاربر را بر عهده دارند. دو مورد از خدمت‌گزارها نقش اساسی دارند؛ یکی از آن‌ها مدیر فرآیند «Process Manager» است که در بر گیرنده آن دسته از فراخوان‌های سیستمی مینیکس۳ است که آغاز یا توقف اجرای فرآیندها را برعهده‌دار دارند، مانند انشعاب «fork»، اجرا «exec» و خروج «exit». هم‌چنین پیاده‌سازی فراخوان‌های سیستمی مرتبط با سیگنالها را شامل می‌شود، نظیر هشدار «alarm» و کشتن یک فرآیند «kill» که قادر هستند وضعیت اجرایی یک فرآیند را تغییر دهند. هم‌چنین مدیر فرآیند در اموری از مدیریت حافظه مانند فراخوان سیستمی «BRK» مسئولیت دارد. دومین خدمت‌گزار، سیستم‌فایل «File System» است که کلیه فراخوان‌های مدیریت حافظه مانند خواندن «read»، اتصال «mount» و تغییر شاخه «chdir» را انجام می‌دهد.

درک تفاوت بین فراخوان‌های هسته و فراخوان‌های سیستمی پاسیکس «POSIX» از اهمیت زیادی برخوردار است. فراخوان‌های هسته عملیات سطح پایینی هستند که به وسیله وظیفه سیستم فراهم شده‌اند تا به گردانندهها و خدمتگزارها اجازه انجام کار خود را بدهند. خواندن یک درگاه ورودی / خروجی سخت‌افزاری، یک فراخوان هسته نوعی به شمار می‌رود. در مقابل، فراخوان‌های سیستمی پاسیکس «POSIX» مانند خواندن «read»، انشعاب «fork» و پیوندزدایی «unlink» فراخوان‌های سطح بالایی هستند که در استاندارد پاسیکس تعریف شده‌اند و برای برنامه‌های کاربر در لایه چهارم، قابل دسترسی می‌باشند.

برنامه‌های کاربر شامل تعداد زیادی از فراخوان‌های پاسیکس است، اما فراخوان‌های هسته ندارد. گاهی در اثر اشتباه لفظی ممکن است فراخوان هسته را فراخوانی سیستمی بنامیم. مکانیسم‌های استفاده شده در ایجاد این فراخوان‌ها مشابه‌اند و فراخوان‌های هسته می‌توانند زیر مجموعه خاصی از فراخوان‌های سیستمی تلقی شوند. همانگونه که اطلاع دارید سیستم‌های عامل دو وظیفه اساسی دارند : مدیریت منابع و فراهم نمودن یک ماشین توسعه یافته به وسیله پیاده‌سازی فراخوان‌های سیستمی. در مینیکس سه مدیریت منابع تا حد زیادی توسط گرداننده های لایه دوم به کمک لایه هسته صورت می‌پذیرد؛ هنگامی که دسترسی ممتاز به درگاه‌های ورودی / خروجی یا سیستم وقفه در خواست می‌شود. تفسیر فراخوان سیستمی به وسیله خدمت‌گزارهای مدیر فرآیند و سیستم‌فایل در لایه سوم، انجام می‌گیرد. سیستم‌فایل دقیقا به صورت یک فایل «خدمت‌گزار» طراحی شده است بطوریکه می‌تواند با تغییرات ناچیزی به یک ماشین راه دور منتقل شود.

در نهایت لایه چهارم متشکل از کلیه فرآیندهای کاربر از قبیل پوسته‌ها، ویرایشگرها، کامپایلرها، و برنامه ساده «Pro-gram.O» ایجاد شده توسط کاربر می‌باشد. بسیاری از فرآیندهای کاربر، با ورود کاربر به سیستم و یا انجام کار و خروج از سیستم، می‌آیند و می‌روند. در مقابل، معمولا در یک سیستم در حال اجرا تعدادی فرآیند کاربر وجود دارند که در زمان راه‌اندازی سیستم آغاز به کار می‌کنند و همیشه در حال اجرا هستند

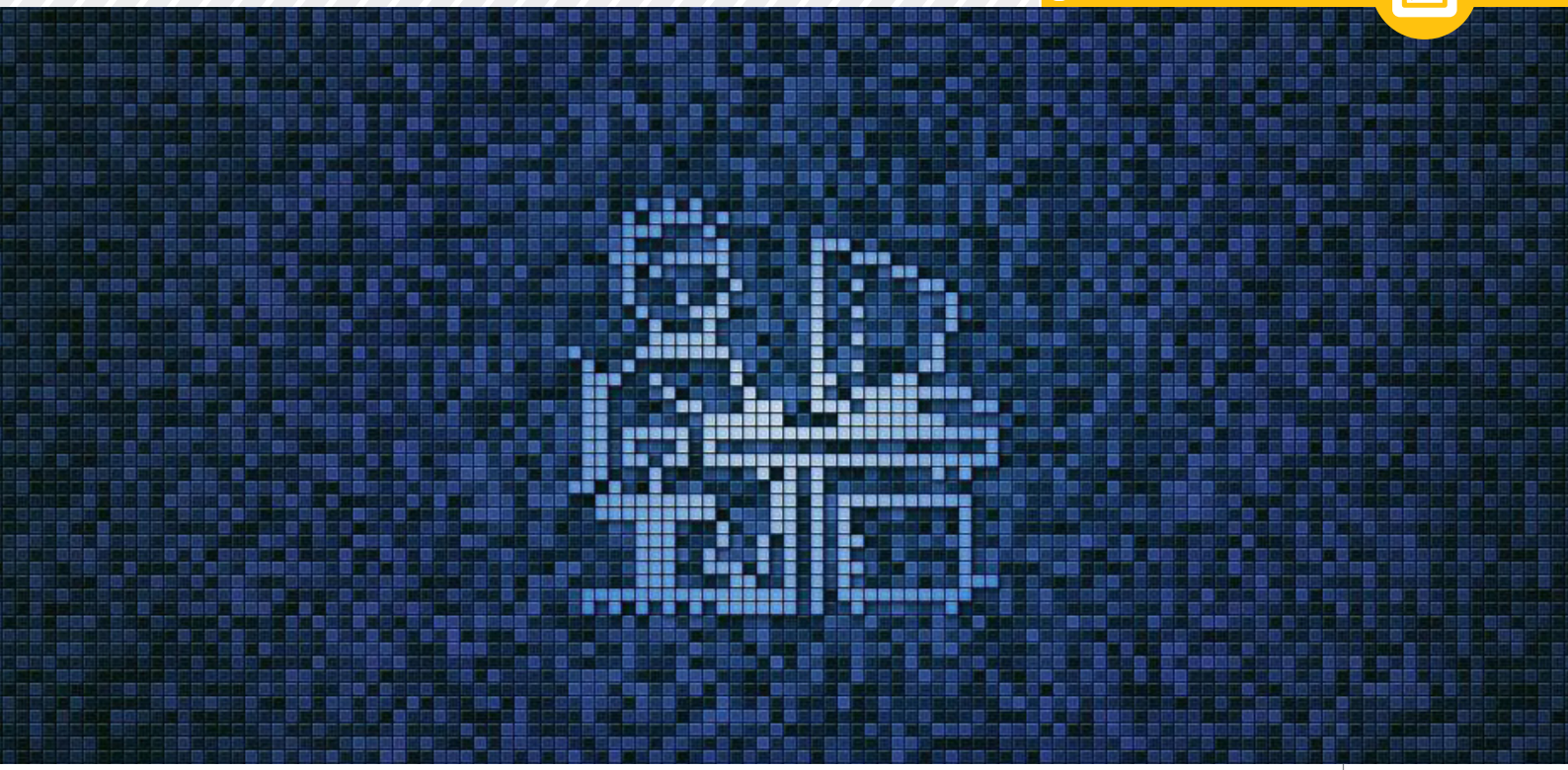
یکی از آن‌ها اینیت «init» است؛ init کوتاه‌شده«Initialization»، نام برنامه یا پروسه‌ای در سیستم‌عامل‌های رایانه‌ای شبه‌یونیکس است که تمام فرآیندهای دیگر را ایجاد می‌کند و بالا می‌آورد. این برنامه به صورت یک خدمت و معمولاً با PID ۱ اجرا می‌شود. راه‌انداز بوت، هسته را شروع می‌کند و هسته نیز اینیت را شروع می‌کند. اگر اینیت را بدون جایگزین کردنش حذف کنید، سیستم در راه‌اندازی مجدد بعدی با «هشدار هسته» مواجه می‌شود. این اینیت است که تعیین می‌کند کامپیوتر چگونه کار می‌کند و آن را هدایت می‌کند،

معمولا چند برنامه به صورت سرویس (دیمن) نیز اجرا می‌شوند. دیمن یا خدمت یک فرآیند پس‌زمینه است که یا به طور متناوب اجرا می‌شود یا همواره در انتظار رویدادی مانند ورود یک بسته از شبکه به سر می‌برد. به عبارتی می توان گفت که این خدمت، خدمت‌گزاری است که به طور مستقل راه‌اندازی می‌شود و به عنوان یک فرآیند کاربر اجرا می‌گردد. خدمت «دیمن» خدمت‌گزارهای واقعی که در هنگام راه‌اندازی سیستم نصب شده‌اند. این امکان وجود دارد که یک دیمن نیز به گونه ای پیکربندی شود که از اولویت بالاتری نسبت به فراخوان‌های معمولی کاربر برخوردار باشد.

دیمن یا خدمت یک فرآیند پس‌زمینه است که یا به طور متناوب اجرا می‌شود یا همواره در انتظار رویدادی مانند ورود یک بسته از شبکه به سر می‌برد. به عبارتی می توان گفت که این خدمت، خدمت‌گزاری است که به طور مستقل راه‌اندازی می‌شود و به عنوان یک فرآیند کاربر اجرا می‌گردد. خدمت «دیمن» خدمت‌گزارهای واقعی که در هنگام راه‌اندازی سیستم نصب شده‌اند

نکته پایانی:

در این‌جا نکته‌ای پیرامون عبارت «وظیفه» و گرداننده دستگاه ضروری به نظر می‌رسد. در نسخه‌های قدیمی مینیکس کلیه گردانندههای دستگاه همراه با هسته ترجمه می‌شدند و امکان دسترسی آن‌ها به ساختارهای داده متعلق به هسته و دیگر وظایف وجود داشت. همچنین آن‌ها می‌توانستند به طور مستقیم به درگاه‌های ورودی/خروجی دسترسی داشته باشند، بنابراین، به آن‌ها تحت عنوان «وظیفه» اشاره می‌شود تا از فرآیندهای مستقلی که به طور محض در فضای کاربر قرار دارند متمایز شوند. در مینیکس سه، گردانندههای دستگاه به طور کامل در فضای کاربر اجرا می‌شوند. تنها استثناء، «وظیفه» ساعت است که از نظر منطقی مانند سایر گردانندههای دستگاه نیست که بتواند از طریق فایل‌های دستگاه به وسیله فرآیندهای کاربر مورد دسترسی قرار بگیرند. ما در درون متن، اصطلاح «وظیفه» را تنها برای «وظیفه» ساعت یا «وظیفه» سیستم به کار بردیم که هر دو برای انجام وظیفه در درون هسته ترجمه شده‌اند. از آن‌جا که نام توابع، نام متغیرها و توضیحات درون کد برنامه بادقت به‌روزرسانی نشده است، شما در طی مطالعه کد برنامه مینیکس سه ممکن است به کلمه «وظیفه» برخورد کنید که در آن‌جا واژه «وظیفه» به معنی گرداننده دستگاه باشد.



ماجراهای سیس‌ادمینی

خلاصه پیش‌گفتار نویسنده:

این نوشته، نسخه ۱.۰ و غیر رسمی از فجایع مدیران سیستم‌عامل یونیکس است که در واقع خلاصه‌ای از همه داستان‌های عنوان شده در گروه خبری comp.unix.ad-min در سال ۱۹۹۲ محسوب می‌شود. من این داستان‌ها را [علیرغم قدیمی بودن] به دو دلیل جمع‌آوری کردم.

۱- بعضی از این داستان‌ها فوق‌العاده جذابند.

۲- افراد زیادی یاد می‌گیرند که وقتی مسئولیت سیستمی به دوششان است، چه زمان نباید چه کاری را انجام بدهند.

این نوشته همه نوع داستانی را در بر می‌گیرد. از داستان‌هایی که پایان موفق و خوبی دارند و... اوممم... داستان‌های دیگر! ولی مهم‌ترین چیزی که آدم‌ها می‌توانند از این داستان‌ها یاد بگیرند این است که «هرگز نشه فراموش، بکاپ بگیر ازشون» (ما همه می‌دانیم که نباید نسخه پشتیبان را فراموش کنیم! مگر نه؟؟). مهم‌تر از گرفتن نسخه پشتیبان این است که مطمئن شویم نسخه پشتیبان کامل و بدون نقص است. برای مثال می‌توانید به داستان گرفتن نسخه پشتیبان یک درایو ۳۰۰ مگابایتی روی نوارهای ۱۵۰ مگابایتی سر بزنید. اگه شما هم خواستید داستان‌هایتان را به این مجموعه اضافه کنید، خوشحال می‌شوم که آن را به آدرس sysadmin@salam-donya.ir بفرستید.

پیش‌گفتار مترجم:

چند روز پیش روز ۲۴ جولای (مصادف با ۲۰۱۵/۰۷/۲۴) بود. اولین جمعه بعد از این روز را روز سیس‌ادمین‌ها نامگذاری کردند. علتش هم گویا این است که سیس‌ادمین‌ها ۲۴ ساعت روز و ۷ روز هفته را مشغول کارند. چند وقت پیش که داشتم در اینترنت و سایت stackoverflow می‌گشتم، مطلبی را با عنوان همین پست دریافت کردم. البته این مربوط به سیس‌ادمین‌های یونیکس بود... و خب، یونیکس و گنو/لینوکس شباهت زیادی دارند. این شباهت که از زمان‌های قدیم استخوان‌بندی این سیستم‌ها را تشکیل داده، باعث می‌شود این داستان‌ها خالی از لطف نباشند. این شما و این هم از داستان‌های فجیع

چند روز پیش روز ۲۴ جولای (مصادف با ۲۰۱۵/۰۷/۲۴) بود. اولین جمعه بعد از این روز را روز سیس‌ادمین‌ها نامگذاری کردند. علتش هم گویا این است که سیس‌ادمین‌ها ۲۴ ساعت روز و ۷ روز هفته را مشغول کارند.

اخطار! خواندن این داستان‌ها برای کسانی که ناراحتی قلبی دارند توصیه نمی‌شود!

از دیو بریلهارت، شرکت هریس سمیکانداکتور (Harris Semiconductor)

خب، اکثرا می‌تونن به این داستان بخندن... اما....

گروه اجرایی ما، یک گروه راهبر با سیستم VMS بود که در تلاش برای یادگیری یونیکس بود. مسئولیت مدیریت اکانت کاربران به این گروه داده شد. آن‌ها شروع کردند به پاکسازی اکانت‌های بلااستفاده از سیستم. وقتی آن‌ها به اکانت sccs برخوردند، دیدند که از زمان تشکیل تا به حال به این اکانت دسترسی انجام نشده. پس آن‌ها از ابزار deluser استفاده کردند که برای پاک کردن همه فایل‌ها و پوشه‌های درون مسیر خانه اکانت، تایید می‌خواهد. به نظر منطقی هست. مگه نه؟؟ (نویسنده: مسیر خانه اکانت sccs در واقع ریشه یا "/" هست. فکر کنم به اندازه کافی گفتم)!

از: دیوید جی داوکینز، دانشگاه ریدینگ:

تقریبا یک سال پیش داشتم داخل پوشه etc/ را می‌گشتم. متوجه شدم که بعضی از فایل‌های داخل این دایرکتوری، دسترسی write عمومی دارند. پس کاری که کردم این

بود که این دستور را وارد کردم:

```
chmod -r 664 /etc
```

همه چیز برای دو الی سه هفته خوب پیش رفت. تا این که ماشین به دلایلی غیر عادی از کار افتاد. روال بوت قادر نبود fsck را اجرا کند. پس فایل‌سیستم‌ها را به صورت فقط خواندنی mount کرد و من را به حالت تک کاربره فرستاد. خشکم زده بود و سرتاپا خیس عرق شدم. مشخص هست که نمی‌توانستم با دستوری مثل chmod +x، چیزی را اجرایی کنم، چون همه چیز فقط خواندنی بود. در واقع دیگر قادر نبودم دستوری را که جزو قسمتی از sh نیست اجرا کنم. زمان سکته کردن فرا رسیده بود.

«چطوره که دیسک را فرمت کنم؟» ذهنم پر از راهکارهای مزخرف شده بود. خودم را مرغ سرکنده‌ای یافتم!

خوشبختانه یک نفر آدم فوق خبره تو یونیکس داشتیم، نشست و بیش از یک ساعت با سیستم کلنجار رفت... و تصمیم گرفت که از cd-rom سیستم را بوت کند... و درستش کرد. هر چند در آن زمان خیلی از سیستم‌های مبتنی بر یونیکس و sun و بالا آمدن سیستم و مدیریت آن‌ها اطلاع نداشتم، اما یک چیز را خوب یاد گرفتم:

هیچ وقت از دستورات به این شکل استفاده نکنم

از: باری اسپنس، شرکت دیتاکد (DataCad):

اشتباه من در سیستم‌عامل Sun (که از Open-Windows استفاده می‌کرد) این بود که سعی کردم تمام فایل‌ها و پوشه‌هایی که با « . » شروع می‌شوند را در مسیر /tmp پاک کنم. بدیهی هست که دستور «/rm -rf /tmp*» این فایل‌ها و پوشه‌ها را نادیده می‌گیرد. پس من بطور خیلی محتاطانه، مطمئن شدم که در مسیر /tmp قرار گرفتم... و دستور «./* rm -rf» را اجرا کردم!

هرگز این کار را دوباره نخواهم کرد! از این به بعد هرگاه در دستوری مطمئن نبودم که یک wildcard تا چه حد می‌تواند گسترش یابد، آن دستور را echo می‌کنم.

[دستور آخر، نه تنها فایل‌ها و پوشه‌هایی که با « . » شروع شده‌اند را شامل می‌شود، بلکه شامل « .. » که به پوشه بالاتر اشاره می‌کند هم می‌شود. کافی هست این مورد را با دستور ls امتحان کنید]

از بروس کاتر، شرکت ای‌تی‌اندتی (AT&T):

مدت‌ها قبل یونیکس System V R4 را روی سیستم ۳۸۶ خانگی‌ام به منظور برنامه‌نویسی نصب کرده بودم. من برنامه‌هایم را در مسیرهای home خودم و همین طور در /usr/local/src کامپایل می‌کردم. پس برای این که فضای دیسک از فایل‌های بیهوده آزاد شود، تصمیم گرفتم که از cron استفاده کنم تا فایل‌های با پسوند «.o» را که در طول روز به کار نمی‌آید را پاک کنم. من دستور زیر را در cron قرار دادم:

```
find / -type f -name \*.o -atime +1 -exec /
;\ {} usr/bin/rm -f
```

به جای آن که از دو دستور زیر استفاده کنم:

```
find /home/bcutter -type f -name \*.o -atime
+1 -exec /usr/bin/rm -f {} \; find /usr/local/
src -type f -name \*.o -atime +1 -exec /usr/
;\ {} bin/rm -f
```

نتیجه این شد که کمی بعد، دیگر قادر نبودم نرم‌افزاری را کامپایل کنم. کاری که خط اول می‌کرد این بود که همه فایل‌های دارای پسوند «.o» را مانند /usr/lib/crt1.o پاک می‌کرد... و بعد فهمیدم که این دستور همه object file‌های مربوط به کرنل را هم پاک کرده است. بعد از این که این اتفاق برای بار دوم و بعد از نصب مجدد سیستم رخ داد، متوجه اشتباه در دستوری که استفاده کرده بودم شدم و آن را اصلاح کردم.

این یک مثال دیگر برای مورد «کار تراشیدن برای جلوگیری از کار بیشتر» است.

هرگز این کار را دوباره نخواهم کرد! از این به بعد هرگاه در دستوری مطمئن نبودم که یک wildcard تا چه حد می‌تواند گسترش یابد، آن دستور را echo می‌کنم.

امنیت ترکیبی از جلوگیری، تشخیص و واکنش است. در حال حاضر، واکنش در بدترین وضعیت ممکن در بین این سه عنصر قرار دارد

واکنش به حملات سایبری مهم‌تر از جلوگیری است

نویسنده:
سعیده‌علی‌محمدی

بروس اشنیر: واحد فناوری اطلاعات سازمان‌ها بیشتر از برنامه جلوگیری از حملات سایبری، نیازمند طرح پاسخ و واکنش به حملات سایبری هستند. واحد فناوری اطلاعات سازمانی و دولتی همواره با شتاب و عجله در صدد جلوگیری از حملات سایبری بزرگ بوده‌اند که اخیرا نیز شرکت‌های بزرگی مثل Sony Pictures، Blue Cross، Anthem، Target، Home Depot و U.S. Department of the Interior با این حملات مواجه شده‌اند. در هر یک از این موارد، هکرها از موقعیت‌های مختلفی در سراسر دنیا به شبکه کامپیوتری حاوی اطلاعات حساس، حساب‌ها و داده‌های شخصی از جمله امنیت اجتماعی و شماره کارت‌های اعتباری مشتریان و کارمندان دسترسی پیدا کرده‌اند. عواقب چنین نفوذهای امنیتی ممکن است بسیار مخرب باشد.

به گفته بروس اشنیر، استاد و متخصص بین‌المللی و بنام در حوزه امنیتی: «همه امیدوارند که نفر بعدی آن‌ها نباشند».بروس اشنیر اضافه کرد، جلوگیری از حملات سایبری تنها بخشی از راه‌حل است. پاسخ یک سازمان به نفوذ امنیتی نیازمند توجه بیشتری است. «ما نیازمند ارائه پاسخ و واکنش بهتر به نفوذ-های امنیتی هستیم. واکنش ما باید، هوشمندانه‌تر، سریع‌تر و کاراتر باشد».

اشنیر در زمینه «حملات، روندها و واکنش‌ها» در LinuxCon، CloudOpen ContainerCon North و America در سیاتل به تاریخ سه‌شنبه ۲۷ مرداد ۹۴ سخنرانی خواهد داشت. در این بخش، اشنیر در زمینه نیاز به یک تغییر مفهومی روی امنیت و آن‌چه سازمان‌ها برای آمادگی بهتر در برابر حملات سازمانی اجتناب‌ناپذیر می‌توانند انجام دهند، بحث خواهد کرد. اشنیر نویسندگی ۱۲ کتاب به همراه صدها مقاله و رساله را به عهده داشته است. ایشان روزنامه معروف «Crypto-Gram» را می‌نویسد و وبلاگ او «Schneier on Security» بیش از ۲۵۰۰۰۰ خواننده دارد. در ادامه مصاحبه کوتاه با ایشان در کنفرانس LinuxCon را مرور می‌کنیم.

Linux.com: به نظر شما امروزه بزرگترین مسئله مفهومی در زمینه امنیت در تکنولوژی چیست؟

بروس اشنیر: به نظر من ما نیازمند یک تغییر مفهومی اساسی درباره نحوه ارتباط یک سازمان با داده هستیم. همواره داده‌ها به صورت مجزا مطرح بوده است که توسط واحد فناوری اطلاعات مدیریت می‌شد. این رویه دیگر قابل اعمال نیست. داده‌ها در مرکز تمامی جوانب یک سازمان قرار دارد و اغلب، مهم‌ترین دارایی هر سازمانی محسوب میشود. با توجه به این اصل، امنیت اطلاعات در اصل امنیت سازمانی است و با توجه به واکنش سمت‌های اجرایی مثل CIO[1] و CISO[2] به این حقیقت، به نظر نمی‌رسد هنوز این مفهوم که داده‌ها بخشی از همه چیز است به قدر کافی جا افتاده باشد.

این بدان معنا است که امنیت اطلاعات، یک مسئله فنی نیست، اگرچه یک مولفه فنی را دربر می‌گیرد و فراتر از آن است. من یک تغییر مفهومی در این مسیر میبینم. گفتگوهای مطرح درباره انعطاف‌پذیری بخشی از آن است، به دلیل این که انعطاف‌پذیری درباره چیزی بیشتر از امنیت فناوری اطلاعات است. انعطاف‌پذیری یک ویژگی اضطراری در نحوه تفکر درباره سازمان‌ها و ریسک و امنیت محسوب می‌شود.

به نظر شما بزرگ‌ترین نکته در مورد حملات در مقیاس بزرگ مثل سونی چیست؟

اشنیر: مهم‌ترین نکته این است که ما همگی در برابر این نوع حملات آسیب‌پذیر هستیم. مهم نیست چه نوع هکرهایی باشند: هکرهای دولتی-ملتی (سونی)، hactivists (HB Gray Federal، Hacking Team)، داخلی (NSA، US State Department) یا who-knows-who (عربستان سعودی)، همگی مستندات داخلی یک سازمان را دزدیده و منتشر می‌کنند که خود این می‌تواند یک حمله مخرب باشد. ما باید در مورد این تاکتیک بیشتر بی‌اندیشیم: کم‌تر «چگونه جلوگیری کنیم»-که هم اکنون در حال انجام است و موثر نیست- و بیشتر «چگونه واکنش نشان دهیم».

در حال حاضر صنعت چگونه با آن برخورد می‌کند؟

اشنیر: هر کسی امیدوار است که نفر بعدی آن‌ها نباشند

مهمترین روشی که سازمانها با آن می-توانند فعالیت‌های امنیتی خود را بهبود بخشند چیست؟

اشنیر: امنیت ترکیبی از جلوگیری، تشخیص و واکنش است. در حال حاضر، واکنش در بدترین وضعیت ممکن در بین این سه عنصر قرار دارد و سازمانها نیازمند بیشترین میزان بهبود در این زمینه هستند. به عبارت ساده‌تر، ما باید در زمینه پاسخ و واکنش به رخدادهای امنیتی ارتقا پیدا کنیم. ما باید هوشمندانه‌تر، سریع‌تر و کاراتر شویم. ما باید پاسخ به رخدادهای حوزه فناوری اطلاعات را با مدیریت بحران سازمان یکپارچه کنیم. ما باید قادر به تشخیص این که چه اتفاقی در سازمان ما در حال حادث شدن است و نحوه برخورد با آن باشیم و باید به نحوی این کار را انجام دهیم که به عنوان یک سازمان، انعطاف‌پذیر باشیم

چگونه می توانیم مسائل امنیتی را در مقیاس جهانی برطرف کنیم؟

اشنیر: اگر من جواب این سوال را میدانستم الان در حال پیاده‌سازی آن بودم. مسائل بین‌المللی بسیار پیچیده هستند و مسائل سایبری مستثنی نیستند. جاسوسی یک مسئله جهانی است. جرایم سایبری، جهانی هستند. نظارت حقوقی شرکت‌های بزرگ یک مسئله جهانی است. در سال‌های آتی این مسائل بسیار بزرگ‌تر و حادتر خواهند شد.

داستان علمی و تخیلی

روزی که مدیر سیستم‌ها بر زمین حکم راندند | ۹۲ |

روزی که مدیر سیستم‌ها بر زمین حکم راندند

مترجم: جلادی میرمیرانی

نویسنده: کروی دکترو

وقتی تلفن مخصوص فلیکس ساعت دو صبح زنگ زد، همسرش جودی غلطی زد و آرام با مشت به شانه او کوبید و گفت «چرا این لعنتی رو قبل این‌که بخوابیم خاموش نکردی؟»

جواب داد «چون آنکال هستم».

روی لبه تخت نشسته بود که جودی با غر و لند گفت «اما تو که دکتر نیستی!» و لگدی هم به سمتش فرستاد. داشت شلوارش را که از روی زمین برداشته بود چپ و راست می‌کرد تا جیبی که موبایل در آن بود را پیدا کند. «آخه مگه تو مدیر سیستم نیستی؟»

- هستم. این هم بخشی از شغل منه.

- ببین دارن مثل خر از تو کار می‌کشن. می‌دونی که حق با منه! آخه چی بگم؟ الان پدری و نمی‌شه که هر وقت نصف شب یکی نتونست اون ویدئوهای لعنتی‌اش رو نگاه کنه، تو بیدار بشی بری استریمش رو راه بندازی. نمی‌شه تلفن رو جواب ندی؟

تلفن هنوز زنگ می‌زد. می‌دانست که حق با همسرش است. تلفن را پیدا کرد و جواب داد. صدایی مکانیکی اعلام می‌کرد «روتر اصلی جواب نمی‌دهد. بی‌جی‌پی جواب نمی‌دهد». صدای مکانیکی ای‌وی‌آر با این‌که او فحش بدهد تا غر بزند هیچ مشکلی نداشت پس حداقل نظرش را اعلام کرد و این کار حس بهتری به او داد.

- شاید هم بتونم از همین‌جا درستش کنم.

می‌توانست از خانه به سیستم‌های برق اضطراری لاگین کند و روترها را ریبوت کند. یوپی‌اس‌ها در شبکه مستقلی بودند که روترهای خودشان را داشتند که به منابع برق بی‌وقفه وصل شده بودند. جودی حالا در تخت نشسته بود؛ مانند شبهی تاریک در جلوی بخش بالایی تخت. «توی این پنج سال زندگی من حتی ندیدم یه بار هم چیزی از خونه درست بشه». البته اشتباه می‌کرد. چیزها بارها از خانه درست شده بودند اما چون این کار بی‌سر و صدا و سریع اتفاق می‌افتاد، کسی آن را به یاد نمی‌آورد. البته شاید هم درست می‌گفت! لاگ‌ها نشان می‌دادند که مشکلاتی که بعد از یک نصفه شب اتفاق می‌افتادند همیشه نیازمند این بودند که فلیکس لباس بپوشد، از خانه بیرون برود، تا شرکت رانندگی کند و مساله را از نزدیک بررسی کند. اسمش را گذاشته بودند قانون انحراف بی‌نهایت جهان یا همان قانون فلیکس.

پنج دقیقه بعد فلیکس پشت فرمان بود. نتوانسته بود مساله را از خانه حل کند. شبکه مستقل روترها هم قطع بود. آخرین باری که این اتفاق افتاد ماجرا مربوط به احمقی بود یک بیل مکانیکی را به کابل اصلی برقی زده بود که دیتاسنتر اصلی سیستم‌های شرکت فلیکس بود. آن ماجرا باعث شده بود او و

تقریبا بیست و پنج مدیر سیستم دیگر یک هفته دائما به کارگاه سر زده بودند تا بالاخره شرکت سرویس‌دهنده و کارگرهای برون‌سپاری شده‌اش توانسته بودند ده هزار سیم قطع شده را دوباره به هم وصل کنند.

موبایلش دو بار دیگر در مسیر زنگ زد تا صدای مکانیکی به روی بلندگوهای بلوتوث منتقل شود و قطعی‌های بیشتری را در قطعات حساس‌تری اعلام کند. یک بار هم جودی زنگ زد. فلیکس گفت:

- سلام

- شاکی نباش. لحنت شاکیه

خندیدن و دوباره گفت:

- لطفا چک کن که توش شاکی بازی نباشه: سلام

همسرش جواب داد:

- عاشقتم فلیکس.

- منم عاشقتم. حسابی. حالا برو تو تخت و بخواب.

- ورژن ۲ بیدار شده!

همیشه این شوخی را می‌کردند. وقتی فرزندانش هنوز در رحم مادر بود می‌گفتند که نسخه بتا است. وقتی کیسه آب پاره شده بود و به دفتر زنگ زده بودند تا خودش را به خانه و بیمارستان برساند همکارش داد کشیده بود «مستر آماده نصبه!» و بعد از این‌که اولین گریه‌اش تمام شد تصمیم گرفتند که فرزندشان به نسخه ۲ رسیده.

تقریبا به دیتاسنتر رسیده بود. گفت «ببخشید که باعث شدم بیدار بشین». خوشبختانه ساعت ۲ صبح از ترافیک خبری نبود. خودرو را کند کرد و به خروجی پیچید و در ورودی پارکینگ ایستاد. نمی‌خواست تماس جودی با رفتن به پارکینگ زیرزمینی قطع شود. در گوشی شنید که:

- مساله بیدار شدن نیست. تو هفت ساله این‌جا کار کردی و سه نفر دارن به تو ریپورت می‌دن. گوشی رو بده به اونها. به اندازه کافی زحمت کشیده‌ای.

جواب داد:

- من دوست ندارم از کارمندهام تسک‌هایی رو بخوام که خودم انجامشون نمی‌دم.

- تو اینکارها رو کرده‌ای! خواهش می‌کنم ازت.. من دوست ندارم شب‌ها تنها باشم.

- جودی..

- من اصلا عصبانی نیستم. فقط جات خالیه. تو که هستی من خیلی خوب می‌خوابم!

- باشه.

- به همین سادگی؟

- بله. به همین سادگی. من دوست ندارم بد بخوابی و به اندازه کافی هم شب‌ها کار کرده‌ام. از حالا فقط تعطیلات آخر هفته‌ها آنکال می‌شم.

جودی خندید و گفت که «سیستم‌ادمین‌ها که تعطیلات آخر هفته ندارن».

- من خواهم داشت. قول می‌دم.

- عالیه. ممنونم.. اوه... ورژن دو روی حوله‌ام کثیف کاری کرد!

- پسر منه دیگه هههه

- بله واقعا هست

و تلفن را قطع کرد. فلیکس خودرو را به حرکت درآورد و پس از توقف کوتاهی جلوی باجه نگهبانی و خیره شدن به داخل اسکنر قرنیه چشم به اندازه‌ای که مطمئن شود دستگاه توانسته چشم خوابالودش را اسکن کند، وارد تونل منتهی به پارکینگ شد.

کنار دستگاه فروش خودکار ایستاد و یک شکولات پاوربار و یک قهوه گرفت. قهوه در لیوان ضدریزش و دارای نی تحویل داده شده؛ لیوانی قابل بردن به اتاق سرورها. پاوربار که شبیه بیسکوییت ولی حاوی مواد مغذی بود را بلعید و جرعه‌ای هم از قهوه نوشید و اجازه داد در ورودی داخلی برای یک لحظه اثر انگشت و ابعاد بدنش را بخواند. در با صدایی خفه باز شد و مثل همیشه هوای خشک دور بدنش را فرا گرفت. بالاخره به داخل اتاق سرورها رسیده بود.

دیوانه خانه‌ای بود. رک‌ها طوری چیده شده بودند که حداکثر یکی دو مدیر سیستم بتوانند لابلایشان حرکت کنند و تمام فضای ارزشمند باقی مانده به رک‌های در حال وز وز اختصاص یافته بود. اما حالا در فضایی به این کوچکی حداقل بیست مدیر سیستم در تقلا بودند. در نگاه اول آدم فکر می‌کرد وارد گردهمایی یک گنگ سیاه پوش شده که علامت مشخصه‌شان تی‌شرت‌های نوشته‌دار و شکم‌های بزرگ و تلفن‌همراه و ابزارهای چندکاره است.

در حالت عادی همه باید در این سالن در حال یخ زدن باشند اما این همه آدم زنده این اتاق کوچک و بدون محفظه را گرم کرده است. پنج شش نفری با وارد شدن او نگاهی به وی انداختند. دو نفر به اسم سلام کردند و او با جواب دادن سعی کرد شکمش را جمع کند و از لابلای آدم‌ها و رک‌ها رد شود و خودش را به بخش آردنت برساند که در انتهای سالن قرار داشت.

ون که حتی امشب آنکال هم نبود گفت «فلیکس!» جواب داد: « تو این‌جا چکار می‌کنی؟ قرار نیست فردا هر دو خسته باشیم و در حال استراحت»

- من؟ اوه... ماشین شخصی خودم حوالی ساعت یک و نیم غیب شد. سیستم مانیتورینگ خبرم کرد. باید زنگ می‌زدم و می‌گفتم من میام این‌جا ولی فکر نمی‌کردم ماشین تو هم خراب شده باشه ماشین شخصی فلیکس که آن را با پنج مدیرسیستم دیگر شریک بود در طبقه پایین بود. با خودش فکر کرد نکند آن هم قطع شده باشد.

- ماجرا چیه؟

- یک حمله فلش‌ورم گسترده! یه عوضی با یه اکسپلویت روز صفر همه ماشین‌های ویندوز روی همه بلوک‌های آی‌پی رو نابود کرده؛ حتی ماشین‌های آی‌پی ورژن شش! همه سیسکوهای بزرگمون روی ورژن شیش هستن و اگر بیشتر از ده تا پروب مونت‌کارلو بهشون برسه پایین می‌یان. در واقع الان همه پایین هستن. دی‌ان‌اس هم مشکل داره! انگار یکی دیشب زون ترنسفر رو آلوده کرده باشه. اوه و البته ایمیل و پیام شخصی‌ها هم هستن. یک بخش از حمله داره تقریبا به هر کسی که توی لیست دوستات هست پیام می‌فرسته که یک تروجان رو نصب و اجرا کنن، اونم با خوندن سابقه ایمیل و چت‌هات با اون آدم و درست کردن یک پیام قانع‌کننده الیزاطوری.

- خدایا!

وان یک مدیر سیستم تیپ ۲ بود. بیشتر از ۱۸۰ قد و موی بلند دم اسبی و سیب آدم برجسته. جواب داده «بله.. دقیقا!»؛ روی تی‌شرتش نوشته بود «اسلحه‌ات رو انتخاب کن» و زیر آن عکس چندین تاس‌بازی‌های ایفای نقش پرینت شده بود. فلیکس تیپ ۱ بود؛ با اضافه وزن در ناحیه کمر و یک ریش کوتاه ولی منظم که غبغبش را می‌پوشاند. تی‌شرت فلیکس می‌گفت «سلام کاتولو» و مزین به یک کاتالوی مهربان و بدون دهن به سبک هِلو کیتی بود. آن‌ها پانزده سالی بود که یکدیگر را می‌شناختند. در یوزنت آشنا شده بودند و بعد حضوری همدیگر را در یک جلسه آبجوخوری فرینت در تورنتو دیده بودند. بعدش هم یکی دو گردهمایی استارترک و در نهایت فلیکس به ون پیشنهاد داده بود که در آردنت با هم کار کنند. ون قابل اعتماد و منظم بود. درس مهندسی برق خوانده بود و یک دفتر سیمی داشت که تمام کارهایی که در مدیریت سیستم می‌کرد را قدم به قدم با تاریخ در آن می‌نوشت.

ون گفت «این بار حتی پیکاک هم نیست». منظورش Problem Exists Between Keyboard And Chair بود یا اشاره به این‌که مشکل از کاربر است - شکلی از مشکلات که اگر آدم‌ها عقل سلیم به خرج دهند پیش نمی‌آید - چیزهایی مثل تروجان‌ها. اما کرم‌هایی که مشغول خوردن روترهای سیسکو بودند مشکل مهندس‌های بی‌سواد بودند.

فلیکس گفت «نه پیکاک نیست. این مشکل مایکروسافت است! هر بار که من ساعت دو شب این‌جا بودم یا مشکل مایکروسافت خبیث بود یا مشکل پیکاک».

در نهایت روترهای موردار را از اینترنت قطع کردند. البته این کار فلیکس نبود چون او دسترسی نداشت روتری را از اینترنت قطع، اینترفیس آی‌پی نسخه شش را خاموش و در نهایت کل دستگاه را ریست کند. دو اپراتور لعنت‌شده از جهنم بودند که باید به شکل همزمان کلیدشان را در قفل‌های قفسه سیسکوها فرو می‌کردند، می‌چرخاندند و به روترهایی که ۹۵٪ ترافیک راه دور کانادا از آن‌ها رد می‌شد دسترسی پیدا می‌کردند.

فلیکس و ون به راحتی ماشین‌های آدرنت را بالا آوردند. ماشین‌ها هنوز تحت حمله مداوم کرم‌ها بودند - روترها بخشی از جریان را بازگردانده بودند. در حال حاضر احتمالا هر ماشینی که در اینترنت بود زیر کرم‌ها غرق شده بود یا خودش مشغول ارسال کرم به جهان بود، شاید هم هر دو.

فلیکس بعد از تحمل صد تایم‌آوت توانست به NIST و باگ‌ترک برسد و پچ‌های کرنل که انتظار می‌رسد فشار کرم‌ها را کمتر کنند دانلود کند و روی ماشین‌های تحت کنترل خودش نصب کند. ساعت ده صبح بود و آن قدر گرسنه بود که می‌توانست یک خرس را برخورد ولی خوشحال بود که توانسته کرنل جدید را کمپایل کند و ماشین‌هایش را مقاوم کند. انگشت‌های بلند ون روی کیبورد مدیریتی می‌چرخید و زبانش وقتی داشت وضعیت لود سیستم‌ها را بررسی می‌کرد بیرون آمده بود.

ون ناراحت بود. گفت «من روی گریدو دویست روز آپتایم داشتم». گریدو قدیمی‌ترین سرور رک بود، تقریبا از همان زمانی که اسم رک را به افتخار کاراکتر جنگ ستارگان نامگذاری کرده بودند. حالا اسم همه بر اساس شخصیت‌های اسمورف‌ها بود. البته در اسمورف‌ها هم تقریبا به آخر اسم‌ها رسیده بودند و مشغول رفتن به سراغ شخصیت‌های مک‌دونالندلند بودند که از لپ‌تاپ ون شروع شده بود. اسم لپ‌تاپ ون میور مک‌چیز بود.

فلیکس گفت «گریدو دوباره برخواهد خواست! اما باورت نمی‌شه. من طبقه پایین یک چهارصد و هشتاد و شیش داشتم که پنج سال آپ‌تایمشه! قلبم می‌شکنه اگه ریبوت شده باشه»

- چهارصد و هشتاد و شش؟! باهاش چیکار می‌کردی آخه؟

- هیچچی. ولی خب کی یه کامپیوتر با پنج سال آپتایم رو شات دان می‌کنه؟ مثل اینه که مادربزرگت رو بکشی چون دیگه کار مفیدی نمی‌کنه.

ون گفت که گرسنه است و باید غذا بخورند.

- خب پس اول ماشین شخصی تو رو بالا میاریم و بعد مال من رو و بعدش یک پیتزا به عنوان صبحانه مهمون منی و بقیه روز رو هم نمیای سر کار.

- قبوله! و رییس تو با ما خیلی خوبی!

ون گفت «تلفنت داره زنگ می‌زنه». فلیکس دستش را از کیس ۴۸۶ بیرون آورد. سیستم در مقابل روشن شدن مقاومت می‌کرد و حالا فلیکس سعی می‌کرد یک پاور جدید که از افراد طبقه بالا قرض گرفته بود را داخل کیس ببندد. ون گوشی را به دست فلیکس داد. گفت «سلام جودی!». صداهای عجیبی در پشت خط شنیده می‌شد. شاید نویز استاتیک. شاید هم نسخه ۲ داشت در حمام دست و پا می‌زد. «جودی؟». خط قطع شد. سعی کرد دوباره زنگ بزند اما نه صدای زنگ شنیده شد نه صدای پیام‌گیر. بعد از چند لحظه تلفن تایم‌آوت داد و پیام «خطا در شبکه» روی صفحه نقش بست. گفت «لعنتی» ولی خشن نبود. تلفن را به کمربندش وصل کرد. جودی می‌خواست بداند چه زمانی برمی‌گردد یا شاید هم لازم داشت در راه برگشت چیزی بخرد. احتمالا پیام می‌گذاشت.

هنوز مشغول سر و کله زدن با پاور جدید بود که تلفنش دوباره زنگ زد. جواب داد و گفت «هی جودی! وضعیت چیه؟». سعی می‌کرد در صدایش هیچ شکلی از غر زدن یا نارضایتی نباشد. احساس گناه می‌کرد: از نظر فنی کار او بعد از بالا آمدن شبکه آردنت تمام شده بود ولی هنوز هم آن‌جا بود. سه سال قبل مشغول کارهای شخصی بود

- البته به هرحال این ساعت‌ها هم در تایم‌شیت حساب می‌شدند و پولش را از کمپانی می‌گرفت.

از تلفن صدای هق هق می‌آمد.

«جودی؟» حس می‌کرد خون بدنش تمام شده و انگشتانش یخ زده‌اند.

صدای جودی ناواضح بود «فلیکس... اون دیگه نفس نمی‌کشه. خدای من. اون مرده!»

- کی؟! کی مرده جودی؟

- ویل

ویل؟ ویل دیگر چه... روی زانوهایش خم شد. ویلیام اسمی بود که در سند تولد نوشته بودند و فراموش کرده بودند. حالا نسخه ۲ نامیده می‌شد. فلیکس ناله‌ای کرد. صدایی شبیه یک سگ بیمار.

- من حالم بده... حتی دیگه نمی‌تونم رو پام بایستم... فلیکس.. من عاشقت هستم.

- جودی؟! چه اتفاقی افتاده؟

- همه، همه! فقط دو تا کانال مونده. خدایا! فلیکس. انگار از پنجره به مرگ خیره شدیم.

صدای استفراغ از تلفن به مغزش هجوم برد. ارتباط داشت قطع می‌شد و حتی صدای استفراغ را هم به سختی می‌شد تشخیص داد. صدا در تلفن می‌چرخید و حالا چیزی به جز صدای خودش نمی‌شنید.

به سمت خط قطع شده فریاد کشید «همونجا باش جودی!» و شماره ۹۱۱ را گرفت اما دوباره پیام «خطا در شبکه» روی گوشی نمایان شد. میور مک‌چیز را از ون گرفت و کابل شبکه ۴۸۶ را به آن زد و فایرفاکس را از کامندلاین اجرا کرد و به دنبال پلیس شهری گوگل کرد. دنبال تماس با ما بود. فلیکس هیچ‌وقت کنترلش را از دست نمی‌داد. کارش حل مساله بود و می‌دانست که وحشت کردن هیچ مساله‌ای را حل نمی‌کند.

یک فرم تماس با ما پیدا کرده بود و مکالمه‌اش با جودی را درست مثل وقتی که مشغول ریپورت کردن یک باگ است در آن وارد کرد. انگشتانش سریع بودند و توضیحاتش دقیق و دگمه *ارسال* را فشار داد.

ون از بالای سرش به صفحه نگاه کرد. گفت «فلیکس..»

فلیکس گفت «خدا!» روی زمین نشسته بود. سعی کرد بلند شود. ون لپ‌تاپ را گرفت و سعی کرد سایت‌های خبری را باز کند اما همه سایت‌ها تایم‌آوت می‌دادند. نمی‌شد گفت مساله به خاطر یک اتفاق عجیب است که همه به سایت‌های

خبـری هجوم برده‌اند یا کار کرمـی که اینترنت را فلج کرده.

فلیکس گفت «باید برسم به خونه».

و ون گفت که رانندگی می‌کند و او را می‌رساند تا بتواند دائما شماره همسرش را بگیرد.

به نزدیکی آسانسورها رسیده بودند. یکی از معدود پنجره‌های ساختمان کنار آسانسور بود. یک پنجره دایره‌ای با شیشـه‌ای بسـیار ضخیم. همان طور که منتظر آسانسـور بودند از آن به بیرون نگاه کردند. به عنوان یک چهارشنبه، ترافیک چندانی به چشم نمی‌خورد. آیا پلیس‌ها بیشتر از معمول بودند؟

ون اشاره کرد و گفت «اوه آسمان‌های بالای سر!» بـرج سـی‌ان از این پنجـره دیده می‌شـد. یک برج مخابراتی عظیم که در شرق آسمان را می‌شکافت. برج کج بود، مثل شـاخه‌ای که در شـن خیس فرو شـده باشـد. آیا واقعا برج داشـت حرکت می‌کرد؟ بله! برج در حال پایین افتادن بود. آرام اما در حال سـرعت گرفتـن. برج داشـت روی منطقـه تجاری سـقوط می‌کرد. یک ثانیه بعد برج از نقطه بحرانی عبور کرد و با خارج شـدن مرکز ثقلش از محدوده پایه برج، ناگهان فرو افتاد. شـوک را حس کردند و بعد صدایش را هم شـنیدند. کل سـاختمان از این سـقوط به خود لرزید. از خرابه‌ها گرد و خاک بلند شده بود و همان طور که بزرگ‌ترین سازه عمودی جهان با شکسـتن سـاختمان‌ها داشت روی زمین آرام می‌گرفت، صداها هم بیشتر می‌شدند.

ون گفت «مرکز مخابرات هم داره می‌ریزه!». نوبت برج‌های شبکه سی‌بی‌سی. بود که به آرامی منحرف شوند. مردم از هر طرف در حال دویدن بودند و آوار بر سـر آن‌ها فرو می‌ریخت. اگر لحظه‌ای تخیلشان را به کار می‌انداختند می‌توانستند تصور کنند که از این پنجره دایره‌ای شکل در حال نگاه کردن به یک تصویر سـاخته شده با کامپیوتر فوق العاده هستند که از تورنت دانلود شده. حالا مدیرسیستم‌ها دیگر هم پشـت پنجره جمع شده بودند و به این سقوط مداوم نگاه می‌کردند.

یکی از آن‌ها پرسید «چی شده؟»

فلیکس گفت «برج سی‌ان سقوط کرد». صدایش از دوردست‌ها به گوش می‌رسید.

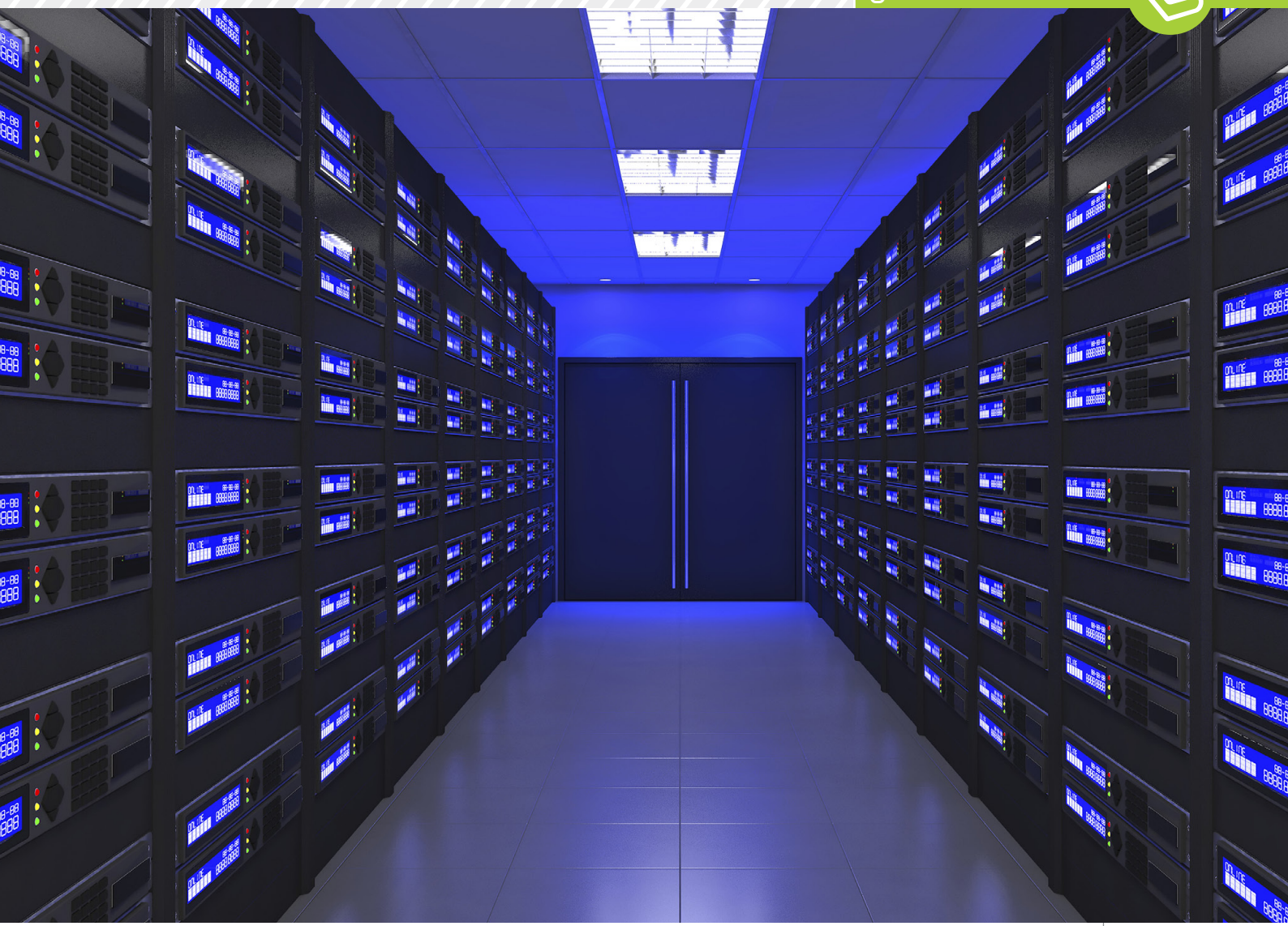
- به این ویروس مربوطه؟

فلیکـس گفـت «به ایـن کرم؟ چی میگـی؟!» و به مرد که مدیر سیستم جوانی بود تازه چربی تیپ ۲ در اطراف شـکمش در حال جمع شـدن بود چشم دوخت.

- نـه کـرم نه! مـن یک ایمیل گرفتـم که می‌گفت شـهر به خاطر یک ویروس قرنطینه شـده. احتمالا یک سلاح بیولوژیک.

مـرد بلک‌بـری اش را بـه فلیکـس داد. فلیکس آن قدر غرق ایمیل فورورد شده از مرکز سلامت کانادا بود که حتی متوجه خاموش شدن تمام چراغ‌های سالن هم نشد. بعد فهمید. بلک‌بری را به صاحبش پس داد و بغض کرد.

یـک دقیقـه بعـد ژنراتورهـا روشـن شـده بودنـد. مدیرسیسـتم‌ها به طـرف پله‌ها دویدند اما فلیکس



بازوی ون را گرفت و او را نگاه داشت.

- به نظرم بهتره ما همینجا منتظر بمونیم

- اما جودی چی می‌شه؟

فلیکس حس کرد دارد بالا می‌آورد.

خودش را جمع کرد و گفت «باید بریم تو اتاق سرورها». اتاق سرورها سیستم تهویه‌ای داشت که هوا از هر غبار و میکروارگانیزمی پاک می‌کرد. بر خلاف اکثریت به سمت پله‌هایی رفتند که به بالا می‌رفت تا خودشان را به بزرگترین اتاق سرور برسانند. فلیکس در را باز کرد و اجازه داد پشت سرش بسته شود.

- فلیکس باید بری خونه

- این یک سلاح بیولوژیکه. سوپرباگ! اینجا سالم می‌مونیم. حداقل تا وقتی فیلترها کار می‌کنن.

- چی؟

- برو ای‌آرسی.

همین کار را کردند. ون پشت میور مک‌چیز بود و فلیکس از اسمورفت استفاده می‌کرد. به چند کانال چت سر زدند تا بالاخره به جایی رسیدند که چندین اسم آشنا مشغول صحبت بودند.

پنتاگون رفته / کاخ سفید هم همین طور

تو سن‌دیگو هستم. همسایه‌هام رو بالکن دارن خون استفراغ می‌کنن

توی گرکین یکی تو خیابون مرده. بانک‌دارها دارن مثل موش فرار می‌کنن

یکی بهم گفته که گینزا آتیش گرفته

فلیکس نوشت: در تورنتو هستیم. خودمون دیدیم که برج سی‌ان سقوط کرد. در مورد سلاح‌های بیولوژیک حرف می‌زنن، یک چیز با اثر خیلی سریع.

ون که نوشته را خوانده بود تذکر داد که نمی‌دانند این مساله چقدر سریع عمل می‌کند و ممکن است همگی مثلا سه روز قبل در معرض آن قرار گرفته باشند. فلیکس جواب داد «اگر این طور بود الان ما هم یه چیزهایی حس می‌کردیم».

به نظر می‌رسه یک پالس قوی الکترومغناطیسی کل هنگ‌کنگ رو گرفته.. شایدم پاریس! هیچ سیگنال شبکه از انجاها نداریم. کل نت بلوک اونها کاملا تاریکه!

تورنتو هستین؟

این شناسه را نمی‌شناختند.

بله. در خیابون فرانت

خواهر من دانشگاه تورنتو است. نمی‌تونم ازش خبر بگیرم. می‌تونین یه زنگ بهش زنین؟

هیچ تلفنی کار نمی‌کنه

فلیکس دوباره به گوشی‌اش نگاه کرد که عبارت «خطا در شبکه» از رویش محو نمی‌شد.

ون با هیجان گفت «من یک سافت‌فون روی میور مک‌چیز دارم!» و برنامه ویپ را اجرا کرد و ادامه داد «اصلا یادم نبود».

فلیکس لپ‌تاپ را قاپید و شماره خانه‌اش را در برنامه تایپ کرد و انتر را فشار داد. تلفن یکبار زنگ زد و بعد تنها چیزی که به گوش می‌رسید یک جیغ ممتد بود. چیزی مانند صدای آمبولانس در فیلم‌های ایتالیایی.

در آی‌ارسی تایپ کرد:

تلفن خط نمی‌ده

و در بالای سرش به ون نگاه کرد و دید که شانه‌های لاغرش می‌لرزند. ون گفت «لعنت تو روح خبیث! دنیا داره تموم می‌شه».

یک ساعت بود که در آی‌آرسی بودند. آتلانتا کاملا سوخته بود. منهتن با حجمی از رادیواکتیو پر شده بود که وبکم‌های نصب شده در لینکلن پلازا دیگر هیچ چیزی مخابره نمی‌کردند. بعضی‌ها مسلمانان تندرو را متهم به عملیات تروریستی می‌کردند اما بعد که مشخص شد به عربستان و نقاط مقدس هم حمله شده، این فرضیه رد شد. دست‌های فلیکس می‌لرزید و ون به آرامی در گوشه اتاق اشک می‌ریخت. دوباره سعی کرد به خانه تلفن کند و بعد هم به پلیس. هیچ فایده‌ای نداشت. درست مانند ۲۰ بار آخری که امتحان کرده بود.

به ماشین طبقه پایین ssh کرد و ایمیل‌هایش را گرفت. اسپم. اسپم. اسپم و بازهم اسپم. ایمیل‌های اتوماتیک. آهان! یک ایمیل از طرف سیستم خودکار تشخیص نفوذ به شبکه آردنت.

ایمیل را باز کرد و سریع خواند. یک نفر با خشونت و به شکل مستمر مشغول چک کردن پورت‌های روترهایش بود. این کار امضای هیچ کرمی را نداشت. آی‌پی را تریس کرد و کشف کرد که حمله از طرف ماشینی در همین ساختمان است - به شکل دقیق‌تر از ماشینی در طبقه پایین.

قدم‌های مواجهه با چنین حمله‌ای از قبل مشخص بود. آی‌پی دستگاه حمله‌کننده را پورت اسکن کرد و مشخص شد که پورت ۱۳۳۷ باز است. این عدد در کارکترهای زبانی/عددی هکرها leet یا elite خوانده می‌شود. این پورت شبیه پورت‌هایی است که کرم‌ها باز می‌گذارند تا بعدا در صورت نیاز از آن داخل شوند. در گوگل به دنبال اسپلویت‌هایی جستجو کرد که از پورت ۱۳۳۷ استفاده می‌کنند و با اضافه کردن اثرانگشت سیستم‌عاملی که این پورت روی آن باز بود به نتیجه رسید. این یک کرم بسیار قدیمی بود. کرمی که احتمالا هر سیستم‌عاملی سال‌ها قبل علیه آن واکسینه شده است. مشکوک بود. کلاینت را دانلود کرد و به ماشین مورد نظر وصل شد و برای خودش یک اکانت روت ساخت. حالا وقتش بود به اطراف نگاهی بیندازد.

یک کاربر دیگر نیز در این سیستم لاگین بود. کاربری به اسم اسکاردی. با دستور تاپ متوجه شد که همین کاربر مسوولیت اجرا کردن صدها پروسه برای چک کردن پورت‌های دستگاه او و تعداد زیادی دستگاه دیگر را بر عهده دارد. یک چت باز کرد و نوشت:

ماشین منو ول کن

انتظار توهین، معذرت خواهی یا اصولا نفی حمله را داشت اما با جواب سورپریز شد:

شما در دیتاسنتر فرانت استریت هستین؟

بله

خدای من. فکر می‌کردم تنها کسی هستم که زنده مونده. من توی طبقه چهارمم. فکر کنم یک حمله بیولوژیک شده. نمی‌خوام از این اتاق تمیز بیرون برم.

فلیکس نفس بلندی کشید و نوشت:

- به ماشینم حمله می‌کردی که من پیدات کنم؟

- آره

- هوشمندانه بوده.

ادمین باهوش خبیث!

ما در طبقه ششم هستیم. دو نفریم.

خبری دارین؟

فلیکس لاگ ای‌آرسی را برایش کپی پیست کرد و منتظر ماند تا طرف مقابل بتواند همه آن را بخواند. ون سر پا بلند شده بود و به اطراف سرک می‌کشید.

- ون؟ رفیق چی شده؟

- باید برم دستشویی.

- در رو باز نکن! یه بطری نوشابه خالی تو سطل آشغال دیدم.

- آره ...

ون مثل یک زامبی به سمت سطلع آشغال رفت و بطری نوشابه خانواده را از آن بیرون کشید و پشتش را به فلیکس کرد.

اسم من فلیکس است.

ویل

معده فلیکس تیر کشید. باز داشت به نسخه ۲ فکر می‌کرد.

ون گفت «فلیکس فکر می‌کنم باید برم بیرون» و به سمت در محفظه هوا راه افتاد. فلکیس کیبورد را به زمین گذاشت و روی پاهایش ایستاد و در تلاش برای حفظ تعادل، به سمت ون دوید و قبل از رسیدن به در او را متوقف کرد.

به چشمان دوستش زل زد. چشمان ون به جای خاصی نگاه نمی‌کردند. گفت «ون! ون! به من نگاه کن»

ون گفت «باید برم. باید برم خونه و به گربه‌هام غذا بدم»

- اون بیرون یک چیزی هست. یک چیز کشنده و خیلی هم سریع. شاید با باد پخش می‌شه. شایدم تا حالا رفته باشه. اما ما تا جایی که بتونیم این‌جا می‌مونیم و فقط وقتی خارج می‌شیم که دیگه هیچ جوری نتونیم این تو بمونیم. بشین ون! بشین.

- سردمه فلیکس

واقعا هم سرد بود. دست‌های فلیکس از سرما مثل پوست مرغ دانه‌دانه شده بودند و پاهایش شبیه قطعات یخ بی‌حس بودند. کلک همیشگی مدیرسیستم‌ها را مطرح کرد: «بیا بریم کنار یک رک داغ. اگر کامپیوتر داخل رک کرم داشته باشه، سی‌پی‌یو داغه و حسابی گرم می‌شیم».

هنوز اونجایین؟

بله. اینجاییم و داریم سعی می‌کنیم یخ نزنیم

چقدر باید صبر کنیم؟

هیچ ایده‌ای نداریم.

تا مدت‌ها کسی چیزی ننوشت.

فلیکس مجبور شده بود دوبار از شیشه نوشابه خانواده استفاده کند. بعد ون از آن استفاده کرد. فلیکس دوباره تلاش کرد به جودی زنگ بزند و سایت پلیس هم هنوز داون بود.

در نهایت کنار رک گرم نشست، با دست‌ها زانوهایش را بغل کرد و مثل بچه‌ها اشک ریخت.

یک دقیقه بعد ون کنارش نشست و دست‌هایش را دور شانه‌هایش حلقه کرد.

فلیکس گفت «ون. همه‌شون مرده‌اند. جودی و پسرم. خانواده‌ام مردن».

- مطمئن نیستیم.

فلیکس غرید «به اندازه کافی مطمئن هستم. همه چیز تموم شده. نشده؟»

- چند ساعت دیگه هم صبر می‌کنیم و بعد می‌زنیم بیرون. تا اون وقت همه چی درست شده. آتشنشانی مشکلات رو حل می‌کنه. ارتش وارد عمل شده. درست می‌شه.

قفسه سینه فلیکس تیر کشید. از تولد نسخه ۲ به بعد گریه نکرده بود. محکم‌تر زانوهایش را بغل کرد.

و بعد درها باز شدند.

دو مدیر سیستم داشتند به داخل سرک می‌کشیدند. یکی از آن‌ها تی‌شرتی داشت که رویش نوشته بود «TALK NERDY TO ME» و نفر دوم تی‌شرت «مدافعان دیجیتال کانادا» به تن داشت.

گیکی حرف بزن گفت: «زود پاشین بیاین. همه تو طبقه آخر جمع شدن. با پله بیاین».

فلیکس فهمید که چند ثانیه‌ای است نفس نکشیده. تـاک نردی گفت: «اگر بیو توی سـاختمون باشـه، همـه آلوده‌ایم. بیاین. اونجـا می‌بینمتون». فلیکس روی پاهایش ایسـتاد و گفت «یکی هم توی طبقه ششـم هست» و جواب شـنید که «آره. اونو دیدیم. رفته بالا».

تاک نردی یکی از مدیرسیستم‌های جهنمی بود که روترهای بزرگ را خاموش کرده بودند. فلیکس و ون به آرامی از پله‌ها بالا می‌رفتند و صدای قدم‌هایشان در راه‌پلـه می‌پیچید. بعد از چندین سـاعت در اتاق سرورها بودن، هوای راه‌پله مانند سونا بود.

در طبقه آخر یک کافه داشـتند و دستشـویی‌هایی کـه به خوبی کار می‌کردنـد. همین طور آب و قهوه و ماشـین خـودکار تحویل خوراکی. صفی نـاآرام از مدیرسیستم‌ها جلوی هر یک از این‌ها تشکیل شده بود. هیچکس به چشم دیگری نگاه نمی‌کرد. فلیکس به این فکر کرد که کدام یک ویل هستند و در صف ماشین خودکار تحویل خوراکی ایستاد. دو انرژی‌بار و یـک لیـوان عظیم قهوه از دسـتگاه گرفـت و پول خردهایش تمام شد. ون روی یکی از میزها برایشان جا گرفته بود. فلیکس یک انرژی‌بار و لیوان قهوه را روی میـز گذاشـت و با گفتن «یـک کمی هم برای من نگه دار» به سـمت صف دستشویی رفت. وقتی خودشـان را خـلاص کرده بودنـد و در حال خوردن بودنـد، تاک نردی و دوسـتش به کافه وارد شـدند. آن‌ها خرت و پرت‌های روی میز پذیرش را به کناری زدنـد و تاک نردی بـه بالای میز رفت و به جمعیت چشم دوخت. صحبت‌ها کم کم قطع شد و سکوت کافه را فرا گرفت.

«مـن یـوری پوپویچ هسـتم و ایـن دیگـو روزنباوم. متشکرم که به این طبقه آمدید. این چیزی است که ما از آن مطمئن هستیم: برق ساختمان سه ساعت است که روی ژنراتور است. شواهد بصری می‌گویند ما تنها ساختمان روشن در تورنتو هستیم و احتمالا تا سـه روز دیگر هم برق خواهیم داشـت. پشـت در این‌جا یک ماده بیولوژیک با منبع ناشـناخته وجود دارد کـه سـریع می‌کشـد؛ در عـرض چند سـاعت. احتمـالا یـک گاز. تنفس هوای آلوده افـراد را آلوده می‌کند. از پنج صبح امروز هیچ کدام از درهای این سـاختمان باز نشده و تا وقتی من تایید نکنم، هیچ کس هیچ دری را باز نخواهد کرد.

«حمله‌هـا بـه شـهرهای بـزرگ در همـه جهـان سیستم‌ها و نیروهای اضطراری را فلج کرده. حمله‌ها الکترونیکی، بیولوژیکی، هسـته و انفجارهای سنتی بوده‌اند و بسـیار گسـترده. من یک مهندس امنیت هستم و این الگوی حمله آشنا است: گروه ب یک پل را منفجر می‌کنند چون همه سیستم‌های محافظتی درگیر حمله اتمی گروه آ هستند. هوشمندانه است. یک هسته آوم شین ریکو در سئول ساعت ۲ بامداد اولیـن حملـه را انجام داده ولی ما مطمئن هسـتیم کـه چنین گروهی نمی‌تواند مسـوول این سـطح از حملات باشد. آن‌ها هرگز به این سلاح‌ها دسترسی نداشـته اند و آن سطح از سازمان‌دهی را هم ندارند که بتوانند در تمام جهان وارد عمل شوند. به عبارت دیگر به اندازه کافی باهوش نیستند.

« ما تا جایی که بتوانیم در این‌جا می‌مانیم، حداقل تا وقتی که سلاح بیولوژیک کشف و خنثی شود. ما تلاش می‌کنیم رک‌ها و سرورها روشن و بالا باشند. این‌جا زیرساخت‌های مهمی داریم و وظیفه ما حفظ

آپتایم «پنج نه» است (اشاره به بالا بودن نود و نه ممیز نهصد و نود و نه درصد مواقع سرورها). در مواقع بحران وظیفه ما چند برابر است»

یکی از مدیر سیستم‌ها دستش را بالا برد. جوان بود و تی‌شرت سبز رنگ هالک به تن داشت.

- کی فوت شده که تو شاه شدی؟

- من کنترل همه سیستم‌های امنیتی رو دارم. همین طور کلید همه اتاق‌های سرور رو و رمز درهای خروجی رو که الان قفل هستن. همچنین من کسی هستم که همه شما رو اینجا جمع کردم و جلسه رو ترتیب دادم. برام مهم نیست اگر یکی دیگه بخواد این مسوولیت رو قبول کنه. کار چرتی است. اما به هرحال یکی باید این کار رو بکنه.

- درست می‌گی. کار بی‌خودیه ولی یکی می‌تونه بکنه. من هم می‌تونم. اسم من ویل ساریو است.

پوپویچ گفت «باشه. اگر بذاری حرف‌هام رو تموم کنم شاید آخرش همه کارها رو تقدیم جنابعالی کردیم».

ساریو پشت به جمعیت کرد و به سمت پنجره رفت و گفت «پس تموم کن» و به بیرون خیره شد. فلیکس خط مسیر او را دنبال کرد. در بیرون از پنجره ستون‌های دود چرب به هوا بلند بود.

کمر سخنرانی پوپویچ شکسته بود. گفت «همین‌ها».

پسرک برگشت و به جمعیت نگاه کرد و گفت «پس حالا نوبت منه؟»

اکثر مدیر سیستم‌ها بدون تمسخر لبخند زدند.

- من این طوری فکر می‌کنم که دنیا داره به گند کشیده می‌شه. حملات هماهنگی به زیرساخت‌های مهم بوده. فقط با یک روش می‌شه این قدر دقیق این همه حمله رو در کل جهان هماهنگ کرد: اینترنت. حتی اگر فکر کنیم حمله‌ها فرصت طلبانه بودن، باید به این فکر کنیم که چطور می‌شه این همه حمله فرصت طلبانه کرد؟ بازم جواب معلومه: اینترنت.

پوپویچ با تمسخر گفت «یعنی می‌گی باید اینترنت رو قطع کنیم؟» و در مقابل جواب ندادن ساریو، خاموش شد.

- ما دیشب یک حمله داشتیم که عملا اینترنت رو قطع کرد. یه داس روی روترهای حیاتی و یک دی‌ان‌اس فو که مثل پنیر همه چیز رو پایین آورد. پلیس و ارتش همیشه از تکنولوژی فراری بودن و این احمق‌ها تقریبا هیچ وقت از اینترنت استفاده نمی‌کنن. اگر اینترنت رو قطع کنیم بیشتر از همه به ضرر حمله کننده‌ها می‌شه و تقریبا روی مدافعین اثری نداره. هر وقت وضع طبیعی شد دوباره وصلش می‌کنیم.»

پوپویچ با دهان باز فریاد زد «مهمل می‌گی!»

ساریو گفت: «منطقی حرف می‌زنم. البته خیلی‌ها وقتی منطق منجر به اقدامات جدی می‌شه، ازش فرار می‌کنن. این مشکل آدم‌ها است نه مشکل منطق».

همه شروع به صحبت با کنار دستی کردند و زمزمه‌ها به سرعت تبدیل به یک همهمه و بعد غرش درهم شد. پوپویچ فریاد زد «خفه شین!» و بحث‌ها یک وات پایین آمد. پوپویچ دوباره فریاد زد و پایش را به میز زیرپایش کوبید و سر و صداها کمی آرام شد. «در هر لحظه یک نفر!». قرمز شده بود ولی دست‌هایش هنوز در جیب‌هایش بود.

یک مدیر سیستم طرفدار ماندن بود، یکی طرفدار رفتن. باید در اتاق سرور پناه می‌گرفتند. باید همه خوراکی‌ها را یکجا جمع می‌کردند و یک نفر مسوول تقسیم می‌شد. باید بیرون می‌رفتند و با پلیس تماس می‌گرفتند یا در بیمارستان‌ها به شکل داوطلب کار می‌کردند. باید کسانی را به حفاظت از درهای ورودی می‌گماشتند. فلیکس از این تعجب کرد که دستش را بالای سرش در حال اجازه گرفتن می‌دید. پوپویچ گفت حرف بزنند.

گفت «اسم من فلیکس ترمونت است» و روی میز رفت و PDAش را بیرون آورد. «می‌خوام براتون یه چیزی بخونم».

«حکومت‌های جهان صنعتی. ای درماندگان گوشت و آهن، من از سایبراسپیس آمده‌ام، از خانه جدید تعقل. من از طرف آینده از شما گذشتگان می‌خواهم که ما را ترک کنید. شما اینجا خوش نیامده‌اید. شما در جهان ما حکمرانی نمی‌کنید.

ما حکومت منتخب نداریم و نمی‌خواهیم هم داشته باشیم پس من با قدرتی بالاتر از آن‌چه لیبرالیسم حکم می‌کند با شما سخن می‌گویم. در فضای اجتماعی جهانی‌ای که من اعلام می‌کنم، قدرت تحمیلی شما جایگاهی ندارد. شما حق اخلاقی ندارید تا بر من حکم برانید و دیگر روشی نیز برای ترساندن واقعی ما در دستانتان نیست.

حکومت‌ها قدرت خود را از پذیرش کسانی می‌گیرند که بر آن‌ها حکم می‌رانند. شما هرگز نظر ما را نپرسیده و از طرف ما پذیرفته نشده‌اید. ما هرگز شما را دعوت نکرده‌ایم. شما حتی ما را نمی‌شناسید، و دنیای ما را نیز. سایبراسپیس خارج از مرزهای شماست. تصور نکنید می‌توانید همانند بقیه پروژه‌های عمرانی‌تان، سایبراسپیس بسازید. نمی‌توانید. این فضا نتیجه طبیعت است و خودش را از طریق اعمال جمعی ما شکل می‌دهد».

سالن ساکت بود. فلیکس گفت «این بخشی از اعلامیه استقلال سایبراسپیس است. این متن ۱۲ سال قبل نوشته شده. من فکر می‌کردم این یکی از زیباترین متن‌هایی است که خوانده‌ام و دوست داشتم کودکانم در دنیایی بزرگ شوند که سایبراسپیس آزاد باشد و آزادی آن بر دنیای فیزیکی هم تاثیربگذارد و گوشت‌اسپیس[1] آزادتر شود».

آب دهانش را قورت داد و با پشت دست چشم‌هایش را پاک کرد. ون به شکلی غریب روی کفش‌هایش زد تا از او تشکر کرده باشد.

«پسر و همسر زیبای من امروز مردند. میلیون‌ها نفر دیگر هم. شهر به معنی واقعی کلمه توی آتیش می‌سوزه. شهرها از نقشه محو شدن».

سرفه‌ای کرد. اشکش را پاک کرد و ادامه داد:

«توی همه دنیا مردمی مثل ما توی ساختمون‌های مثل این جمع شدن. همه داشتن سعی می‌کردن سرورها رو به اینترنت برگردونن و سیستم‌هاشون رو از دست اون کرم نجات بدن. ما غذا و آب و برق داریم. ما شبکه‌ای داریم که آدم‌های بد ازش این قدر خوب استفاده کردن و آدم‌های خوب چندان ازش سر در نیاوردن. ما همه علاقه مشترکی به آزادی داریم که ریشه‌ای عشق ورزیدن به شبکه اینترنته. ما مسوول این مهم‌ترین سازمان و ابزار هستیم که دنیا تا به حال به خودش دیده. ما در حال حاضر نزدیک‌ترین چیز به مفهوم دولت هستیم. ژنو از بین رفته، رود شرقی تو آتیش می‌سوزه و سازمان ملل هم که تخلیه شده.

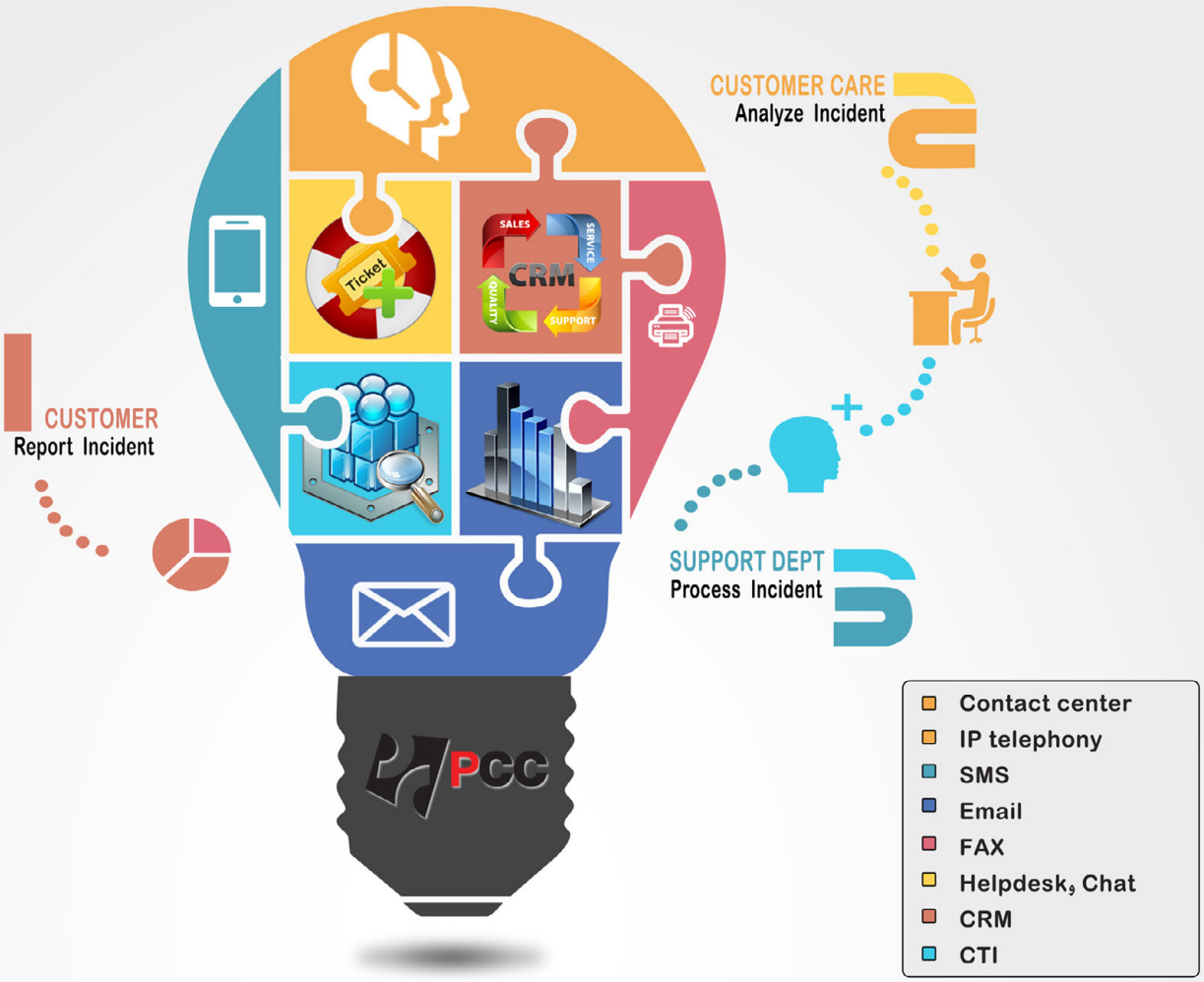
«جمهوری توزیع شده سایبراسپیس این طوفان رو بدون صدمه رد کرده. ما متولی این ماشین بدون مرگ، عظیم و شگفت‌آوریم. ماشینی که می‌تونه دنیا رو دوباره شکل بده. من به جز این چیزی برای زنده موندن ندارم».

چشم‌های ون پر از اشک شده بود. او تنها نبود. کسی او را تشویق نکرد اما اتفاق بهتری افتاد. همه با احترام سکوت کردند. سکوت کامل همراه با فکری که تا یک دقیقه به طول کشید.

پوپویچ گفت: «چطوری باید این کار رو بکنیم؟». در صدایش هیچ رگه‌ای از استهزا نبود.■

ادامه داستان در شماره بعدی

۱. Meat Spaice
که منظور دنیای واقعی ما انسان‌هاست



PCC برخی از امکانات سیستم

توزیع خودکار تماس ها (ACD)
امکان تعریف صف های پاسخگویی به هر تعداد
امکان تعریف وزن برای صف
امکان تعریف وزن برای اپراتورها
امکان تعریف منوی صوتی تعاملی (IVR)به صورت چند لایه ای
امکان تعریف شرایط زمانی
امکان تعریف کنفرانس تلفنی و ویدوئی
امکان انتقال مکالمات(Call Transfer)
امکان تعریف تعداد نامحدود داخلی
امکان تعریف اپراتور به همراه نام کاربری و کلمه عبور برای ورود به صف
امکان ورود به همه صف های دارای دسترسی با گرفتن کد ویژه
امکان تعریف داخلی های دارای دسترسی به صورت ریموت (پاسخگوئی از راه دور)
امکان تعریف لیست سیاه شماره ها
امکان محدود کردن ضبط مکالمات برای داخلی ها و یا صف های مورد نظر
امکان ضبط کلیه مکالمات و نگهداری آنها
امکان تعریف نظرسنجی
امکان تعریف صندوق صوتی
امکان هدایت تماس ها بر اساس شماره تماس گیرنده
امکان ایجاد درخواست تلفنی و ثبت آن در سیستم پشتیبانی

ParsPooyesh
Fanavar
www.parspooyesh.com
info@parspooyesh.com
+982148056000
+982148056789

متنک

می توانید هر خبر یا مقاله ای را خلاصه کنید

متنک آماده شده شما برای آدرس: http://matnak.com

با توجه به حجم انبوه اطلاعات، دسترسی به داده های مطلوب گاها بسیار پیچیده می شود. متنک خدمتی است که می توانید از آن برای سهولت دسترسی به اطلاعات مورد نظر استفاده کنید. از هر خبر منتشر شده، قسمت های مهم آن خبر را بیابید و طبقه بندی کنید و یا نتایج موجود در تحقیق و یا مقاله ای را جمع بندی نمائید.

در نهایت شما قادر خواهید بود متنی را به عنوان ورودی به متنک ارائه دهید و آن را خلاصه کنید.

روان ارتباط

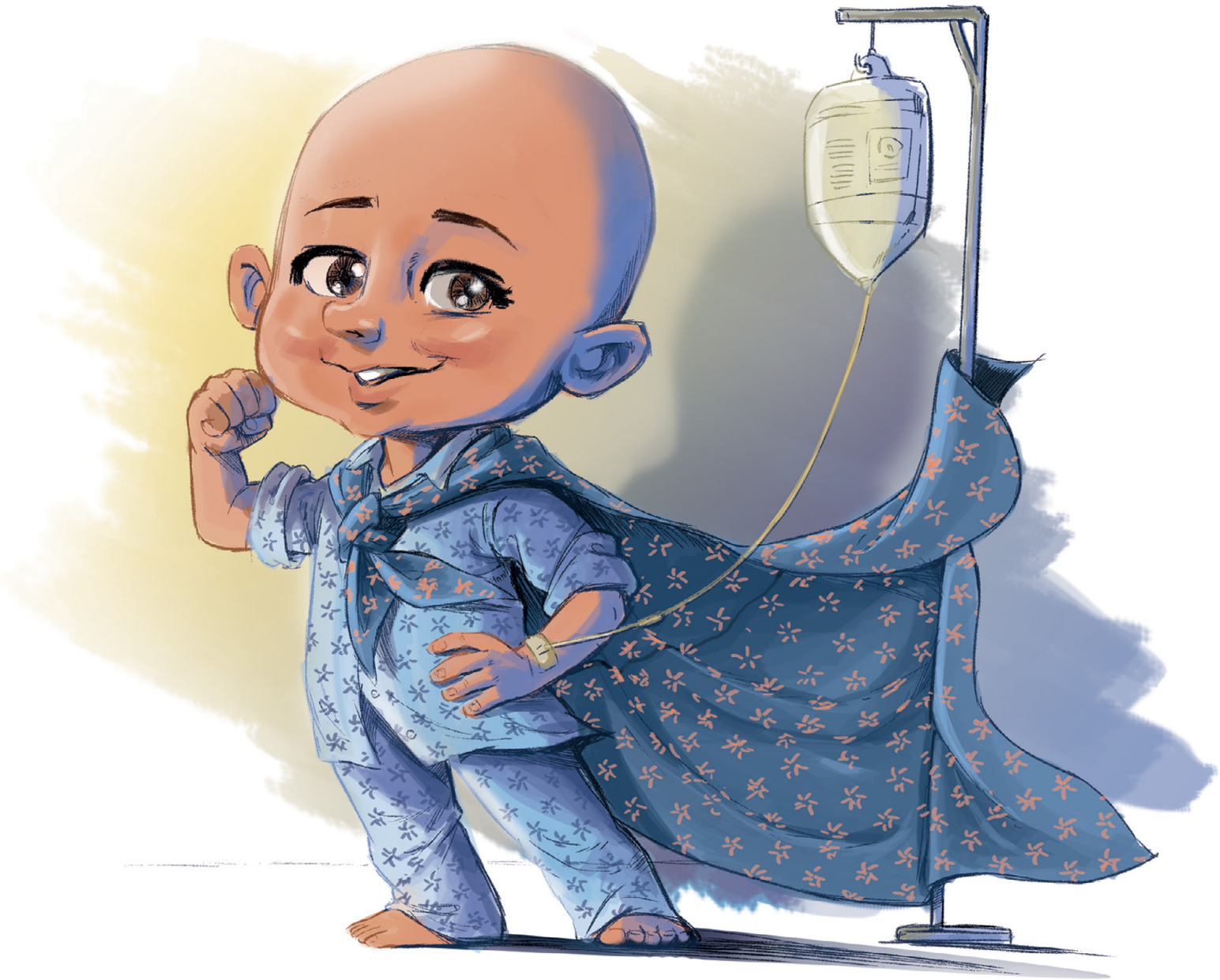
سیوس
راه کارهای بر پایه وب

ایران، تهران، میدان محسنی، بلوار میرداماد، جنب بانک آینده، ساختمان کامیار، طبقه سوم، واحد ۱۸

تلفن: (داخلی ۱۱۱) ۲۲۹۰۳۹۲۴ ۲۱ (۹۸+)

نمابر: ۲۲۲۲۹۷۵۶ ۲۱ (۹۸+)

www.ravanertebat.com
www.cvas.ir



سـرطان را شکست دهیم

در این روزها که دل هایمان پر از امید است و هر روز خبر قهرمانی و پیروزی ایرانیان را می شنویم،
کمک کنیم تا کودکان محک نیز سرطان را شکست دهند و قهرمانی خود را جشن بگیرند.

برای پیوستن به خانواده بزرگ محک و آگاهی از روش های کمک به کودکان مبتلا به سرطان با ما تماس بگیرید.

محک
مـؤسسه خیریه حمایت از
کـودکان مبتلا به سرطان
mahak-charity.org

۰۲۱ – ۲۳۵۴۰ ☎ | #۲۳۵۴۰*۷۲۰* 📱

شماره حساب بانک پارسیان: ۸۱۰۴۴۴۴۹

از اینکه به پیام ما توجه کردید، سـپاسگزاریم.