لینوکس و زندگی

جادی میرمیرانی

۲۷ فروردین ۱۳۹۸

فهرست مطالب

درباره کتاب

این کتاب قرار بود کتابی باشه در مورد لینوکس با اشاره‌هایی به زندگی گیک های جوان. ولی در واقع به این نتیجه رسیده ام که در این کتاب حرف زدن از لینوکس اونقدر که حرف زدن از زندگی مهمه، مهم نیست. ما کتاب های زیادی در مورد لینوکس داریم ولی جاهای کمی هستن که از زندگی حرف بزنن. اینه که این کتاب به زودی به کتاب زندگی گیک‌های جوان تبدیل خواهد شد.

فصل ۱

اصول و مقدمات

۱.۱ فلسفه آزادی نرم‌افزار

نرم‌افزار آزاد نرم‌افزاری است که می‌توان آن را آزادانه و بدون محدودیت، به هر منظور استفاده کرد، مطالعه و بررسی نمود، و تغییر داد. همچنین کپی کردن یا توزیع مجدد (خواه بدون تغییر و خواه با تغییراتی در نرم‌افزار) آزاد و بدون محدودیت یا با محدودیت بسیار کمی (تنها برای اطمینان از اینکه دریافت کنندگان بعدی نرم‌افزار نیز از این آزادی‌ها بهره‌مند می‌شوند یا تولیدکنندگان سخت‌افزارهایی که سروکار سخت‌افزار با مصرف‌کننده است به کاربران اجازه‌ی ایجاد تغییر در سخت‌افزارشان را بدهند) است. نرم‌افزارهای آزاد عموما رایگان هستند اما می‌توانند دارای قیمت هم باشند مثلا برای هزینه تولید CD و دیگر اَشکال توزیع آن.

در عمل، کد مبدا نرم‌افزارهای آزاد همراه با یادداشتی که آزادی‌های بالا را تامین می‌کند عرضه می‌شود که به آن اجازه‌نامه نرم‌افزار آزاد گفته می‌شود.

جنبش نرم‌افزار آزاد در سال ۱۹۸۳ میلادی به پیشگامی ریچارد استالمن به راه افتاد تا نیاز کاربران کامپیوتر به مزایای آزادی نرم‌افزار را تامین کند. استالمن بنیاد نرم‌افزار آزاد را در ۱۹۸۵ میلادی برای تامین ساختار سازمانی لازم برای پیشبرد ایده‌های نرم‌افزار آزادش تاسیس کرد.

تعریف

نرم‌افزاری که آزادی‌های زیر را برای کاربر قائل شود، نرم‌افزار آزاد خوانده می‌شود (توجه کنید که کلمه Free به معنای آزاد استفاده می‌شود و نه رایگان!):

۰. آزادی اجرای برنامه برای هر کاری (آزادی صفرم)

۱. آزادی مطالعه چگونگی کار برنامه و تغییر آن (پیش نیاز: متن برنامه) (آزادی یکم)

۲. آزادی تکثیر و کپی برنامه (آزادی دوم)

۳. آزادی تقویت و بهتر کردن برنامه و توزیع آن برای همگان (پیش نیاز: متن برنامه) (آزادی سوم)

هر نرم‌افزار آزاد، چنین آزادی‌هایی را برای کاربر دارد. علاوه بر این‌ها، یک شرط هم هست و آن هم این هست که اگر شما از این آزادی‌ها استفاده کردید و نرم‌افزاری را تولید کردید و آن را به دیگران دادید، باید این آزادی‌ها را به کاربران‌تان هم بدهید. اگر شما این آزادی‌ها را داشتید پس دیگران هم باید داشته باشند، یعنی نرم‌افزار آزاد تا آخرین توزیعش باید آزاد بماند.

آزادی نرم‌افزارهای آزاد تا جایی هست که حتی می‌توان بدون پرداخت هزینه‌ای برای مجوز، کپی‌هایی از

یک نرم‌افزار آزاد را، یا بدون تغییرات، رایگان یا در ازای دریافت وجه، برای هرکس و هرجایی آن را توزیع کرد.

نرم‌افزارهای آزاد (به دلیل ابهام در لفظ Free) به اشتباه به‌عنوان نرم‌افزارهای رایگان و احتمالاً بی‌ارزش تلقی می‌شدند، به همین دلیل این نرم‌افزارها به متن باز یا متن آزاد (Open Source) معروف شدند. در واقع در نرم افزارهای آزاد قیمت مورد نظر نیست بلکه آزادی مطرح است.

از دیگر ضمانت‌هایی که نرم‌افزار آزاد تأمین می‌کند، اجازه‌نامه عمومی همگانی (GPL) است. GPL برای هر کس امکان دوباره توزیع‌کردن یا کامپایل مجدد متن برنامه را فراهم می‌کند. طبق این اجازه‌نامه باید متن برنامه در دسترس قرار داده شود تا امکان استفاده و یا تغییر آن باشد. برنامه‌های رایانه‌ای اینگونه را معمولاً متن باز گویند. متن چنین برنامه‌هایی نمی‌تواند به حالت «محدود شده» درآید مگر با نظر تک تک نویسندگان آن متن. بیشتر نویسندگان متن لینوکس تحت این مجوز برنامه‌نویسی می‌کنند.

انگیزه

از انگیزه‌هایی که باعث ایجاد نرم‌افزارهای آزاد شد می‌توان رقابت نرم‌افزارهای آزاد و سرمایه‌گرایی را ذکر کرد. فعالان این جنبش معتقدند که محدودیت‌هایی که سرمایه‌گرایی به نرم‌افزارها اعمال می‌کند، مانع از اصلاح و پیشرفت فنی آنها می‌شود و با این نوع محدودیت‌ها مخالفند.

حقوق پدیدآورنده

مسلماً اختراع یک نرم‌افزار حقوق مادی و معنوی برای مخترع نرم‌افزار ایجاد می‌کند که در ایران تحت عنوان قانون حمایت از حقوق پدیدآورندگان نرم‌افزارهای رایانه‌ای به تصویب رسیده است.

از جمله حقوق معنوی می‌توان به موارد زیر اشاره کرد:

- حق انتساب (نام پدید آورنده ذکر شود)
- حق یکپارچگی اثر
- حق انتشار گمنام یا نام مستعار
- و از جمله حقوق مادی می‌توان به حق تغییر یا نشر با اجازه پدیدآورنده اشاره کرد.

کپی‌لفت

شما اجازه ندارید با افزودن محدودیت‌هایی به یک نرم‌افزار تحت حمایت قانون کپی‌لفت، آزادی‌های مرکزی آن را برای دیگران از بین ببرید. این قانون نه تنها با آزادی‌های مرکزی در تضاد نیست بلکه از آنها محافظت می‌کند.

برای این نرم‌افزارها اجازه‌نامه قابل قبول است که اگر یک نسخه‌ی تغییر یافته از برنامه را توزیع کردید و توسعه‌دهنده‌ی قبلی یک کپی از آن را درخواست نمود، شما باید یک کپی برای او بفرستید.

امنیت

نرم‌افزارهای آزاد معمولاً با سرعت بیشتری نسبت به نرم‌افزارهای انحصار گرایانه به‌روز می‌شوند و حفره‌های امنیتی که در نسخه‌های پیشین وجود داشته، در نسخه‌های جدید اصلاح می‌شود.

مثالهایی از نرم‌افزارهای آزاد کاربردی

- هسته سیستم‌عامل گنو/لینوکس، داروین (هسته‌ی مک) و بی‌اس‌دی
- کامپایلر جی‌سی‌سی، کتابخانه‌ی C
- پایگاه‌داده‌های رابطه‌ای مانند: MySQL ، PostgreSQL
- زبان‌های برنامه‌نویسی مانند تی‌سی‌ال، روبی، پایتون، پرل و پی‌اچ‌پی.
- مرورگر وب: فایرفاکس اُپن آفیس
- میزکار کی‌دی‌ای
- میزکار گنوم
- برنامه‌های حروف چینی مانند تک، لاتک و فارسی تک[1]
- نرم‌افزارهای مدیریت محتوا: دروپال، جوملا، پی‌اچ‌پی نیوک، پست نیوک و مامبو.
- نرم‌افزارهای ساخت انجمن: phpBB

[1] این پی دی اف نیز با لاتک و بسته زی پرشن ساخته شده است :)

۲.۱ تاریخچه گنو/لینوکس

گنو

گنو (GNU) یک سیستم‌عامل آزاد شبه‌یونیکس است که توسط پروژه‌ی گنو توسعه پیدا میکند. گنو مخفف "گنو یونیکس نیست" (GNU's Not Unix) است و این نام بخاطر این انتخاب‌شده که اولاً طراحی گنو، شبه‌یونیکس است و ثانیاً گنو جزء نرم‌افزارهای آزاد بوده و از کدهای یونیکس استفاده نمیکند.

ریچارد استالمن موسس بنیاد نرم‌افزار آزاد، کار خود در دانشگاه MIT را در سال ۱۹۷۱ آغاز کرد. در آن زمان نرم‌افزارهای آزاد، همکاری برنامه‌نویسان و کاربران، به اشتراک‌گذاری کد و ... رونق داشت. اما در دهه‌ی ۸۰ انحصار و مالکیت بر نرم‌افزارها، عدم قبول همکاری کاربران در گسترش نرم‌افزارها و بطور خلاصه تجاری‌شدن نرم‌افزارها شدت گرفت. پروژه‌ی گنو جنبشی بر علیه محدودیت‌ها و موانع اعمال‌شده توسط صاحبان نرم‌افزارهای انحصاری و با هدف طراحی نرم‌افزار آزاد بود.

قدم اول در این راه ایجاد یک سیستم‌عامل آزاد بود. طراحی سیستم‌عامل گنو توسط ریچارد استالمن در سال ۱۹۸۳ آغاز شد. استالمن همچنین در سال ۱۹۸۵ بنیاد نرم‌افزار آزاد را بیشتر با هدف جذب سرمایه برای توسعه‌ی گنو تاسیس کرد. در ابتدا اجزاء مورد نیاز هسته گنو مثل: ویرایشگرها، پوسته‌ها، کامپایلرها و سایر ابزارها طراحی و پیاده‌سازی شدند اما هسته‌ی سیستم‌عامل هنوز مهیا نبود. هسته‌ی گنو، هرد (Hurd) نام دارد و از سال ۱۹۹۰ تاکنون در دست توسعه است. با این وجود هسته‌های غیر گنو که معروف‌ترین آنها لینوکس[1] است میتوانند با نرم‌افزارهای آزاد گنو کار کنند. سیستم‌عامل گنو/لینوکس محصول ترکیب هسته‌ی لینوکس و نرم‌افزارهای آزاد گنو است.

در وب‌سایت اختصاصی پروژه‌ی گنو هدف نهایی این پروژه بدین شکل بیان شده است:

> پروژه‌ی گنو فقط به یک سیستم‌عامل محدود نشده است. ما در نظر داریم تا یک مجموعه کامل از نرم‌افزارها را ایجاد کنیم، هر آنچه که بسیاری از کاربران میخواهند داشته باشند. هدف نهایی فراهم‌کردن نرم‌افزارهای آزاد برای انجام تمام کارهایی که کاربران کامپیوتر میخواهند انجام دهند و در نتیجه مطرودکردن نرم‌افزارهای انحصاری است.

یونیکس

به منظور درک محبوبیت لینوکس باید سفری به زمان گذشته داشته باشیم، در حدود ۳۰ سال پیش ...

کامپیوترها را به اندازه‌ی خانه‌ها تجسم کنید، حتی به اندازه‌ی استادیوم‌ها. علاوه بر اینکه اندازه‌ی آن کامپیوترها مشکلات قابل توجهی بوجود می‌آورد مسئله دیگری نیز این را بدتر میکرد: هر کامپیوتر سیستم

[1] لینوس توروالدز هسته‌ی لینوکس را در سال ۱۹۹۱ نوشت و آنرا تحت مجوز GPL منتشر کرد.

عامل مجزایی داشت. نرم‌افزار، برای برآورده‌کردن یک نیاز خاص سفارشی میشد و نرم‌افزار روی یک سیستم، بر روی سیستم دیگری اجرا نمیشد. قابلیت کار کردن با یک سیستم به این معنی نبود که شما میتوانید با دیگری هم کار کنید. این قضیه هم برای کاربران و هم برای مدیران سیستم دشوار بود. کامپیوترها بینهایت گران بودند و حتی پس از خرید اصلی باید تلاش‌هایی در جهت اینکه کاربران بفهمند آنها چگونه کار میکنند صورت میگرفت. کل هزینه بر مبنای واحد قدرت محاسباتی بسیار هنگفت بود. فناوری جهان نسبتاً پیشرفته نبود بنابراین آنها مجبور بودند با این وضع برای یک دهه‌ی دیگر کنار بیایند.

در سال ۱۹۶۹ یک تیم از توسعه‌دهندگان در آزمایشگاه‌های بل روی راه‌حلی برای معضل نرم‌افزار جهت درست‌کردن اینگونه مشکلات سازگاری شروع به کار کردند. آنها سیستم عامل جدیدی را توسعه دادند که:

۱. ساده و دلپسند بود.

۲. به جای کد اسمبلی با زبان برنامه‌نویسی سی نوشته شده بود.

۳. قادر به بازیافت کد بود.

توسعه‌دهندگان آزمایشگاه‌های بل نام پروژه‌شان را یونیکس (`Unix`) گذاشتند. ویژگی‌های بازیافت کد بسیار مهم بودند. تا آن زمان همه‌ی سیستم‌های کامپیوتری تجاری موجود با کدی نوشته شده بودند که به طور خاص برای یک سیستم توسعه داده شده بود. از طرف دیگر یونیکس فقط به تکه‌ی کوچکی از آن کد بخصوص نیاز داشت که امروزه عموماً هسته نامیده میشود. این هسته تنها تکه کدی است که برای انطباق با هر سیستم بخصوص و شکل‌دادن به پایه‌ی سیستم یونیکس مورد نیاز است. سیستم عامل و همه‌ی کارکردهای دیگر، حول این هسته ساخته و در زبان برنامه‌نویسی سطح بالاتر سی نوشته شده‌اند. این زبان به طور ویژه برای ایجاد سیستم یونیکس توسعه داده شد. با استفاده از این تکنیک جدید توسعه‌ی سیستم عاملی که بتواند روی سخت‌افزارهای مختلف اجرا شود بسیار آسان‌تر شد.

فروشندگان نرم‌افزار به سرعت وفق پیدا کردند، چون میتوانستند ده‌ها بار بیشتر نرم‌افزاری که تقریباً بی‌دردسر بود را بفروشند. موقعیت‌های شگفت‌انگیزی بوجود آمد: تصور اینکه برای نمونه کامپیوترهای فروشندگان مختلف در یک شبکه‌ی واحد ارتباط برقرار کنند، یا کاربرانی که بدون اینکه نیاز داشته باشند برای استفاده از کامپیوتر دیگر آموزش اضافی ببینند، روی سیستم‌های گوناگون کار میکنند. یونیکس سهم بسزایی در جهت کمک به کاربران برای سازگاری با سیستم‌های مختلف ایفا کرد.

در طول چند دهه‌ی آینده توسعه‌ی یونیکس ادامه پیدا کرد. انجام خیلی از چیزها امکان‌پذیر شد و بسیاری از فروشندگان سخت‌افزار و نرم‌افزار، پشتیبانی از یونیکس را برای محصولات‌شان اضافه کردند.

یونیکس در ابتدا تنها در محیط‌های خیلی وسیع با مین‌فریم‌ها و مینی‌کامپیوترها مستقر شد[۱]. شما مجبور

[۱] توجه داشته باشید که پی‌سی یک میکروکامپیوتر است.

بودید برای کار کردن با یک سیستم یونیکس، یا در یک دانشگاه، یا برای دولت و یا برای موسسات مالی بزرگ کار کنید.

اما کامپیوترهای کوچکتر توسعه پیدا کردند و در اواخر دهه‌ی ۸۰ بسیاری از مردم کامپیوتر خانگی داشتند. در آن زمان چندین نسخه از یونیکس برای معماری پی‌سی موجود بود اما هیچ‌کدام از آنها واقعا آزاد و مهمتر از آن سریع نبودند، همه‌ی آنها بطور وحشتناکی کند بودند، بنابراین خیلی از مردم با MS DOS یا Windows 3.1 روی کامپیوترهای خانگی‌شان کار میکردند.

لینوس و لینوکس

با آغاز دهه‌ی ۹۰ کامپیوترهای خانگی سرانجام به اندازه‌ای قدرتمند شدند که بتوانند یک یونیکس تمام‌عیار را اجرا کنند. لینوس تورالدز، جوانی که در دانشگاه هلسینکی علوم کامپیوتر میخواند، فکر کرد ایده‌ی خوبی است که یک جور نسخه‌ی دانشگاهی آزادانه در دسترس یونیکس را داشته باشد و بی‌درنگ شروع به کدنویسی کرد. او شروع به سوال‌پرسیدن، یافتن جواب‌ها و راه‌حل‌هایی که میتوانست به او در داشتن یونیکس روی پی‌سی‌اش کمک کند، کرد.

در زیر یکی از اولین پست‌های او در comp.os.minix مربوط به سال ۱۹۹۱ را مشاهده میکنید:

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict
Torvalds)
Newsgroups: comp.os.minix
Subject: Gcc-1.40 and a posix-question
Message-ID: 1991Jul3.100050.9886@klaava.Helsinki.FI
Date: 3 Jul 91 10:00:50 GMT
```

سلام شبکه‌ای‌ها،
به خاطر پروژه‌ای که مشغول آن هستم (در مینیکس)، علاقمندم تعاریف استاندارد پوسیکس را داشته باشم. ممکن است یک نفر من را به یک نسخه (ترجیحا) قابل خواندن توسط ماشین از جدیدترین نسخه راهنمایی کند؟ سایت‌های اف.تی.پی. خیلی خوب خواهند بود.

از همان آغاز هدف لینوس داشتن یک سیستم آزاد که با یونیکس اصلی سازگار باشد بود. به همین علت او در مورد استانداردهای POSIX پرسید. POSIX هنوز هم استاندارد یونیکس است.

آن روزها نصب و اجرا[۱] هنوز اختراع نشده بود اما خیلی از مردم علاقه‌مند به داشتن یک سیستم یونیکس

[۱] plug-and-play

برای خودشان بودند که این فقط یک مشکل کوچک بود. درایورهای جدید با شتاب تندی برای همه نوع سخت‌افزار جدید در دسترس قرار گرفتند. تقریباً به محض اینکه قطعه‌ی سخت‌افزاری جدیدی در دسترس قرار میگرفت شخصی آنرا میخرید و جهت تست لینوکس ارائه میکرد.

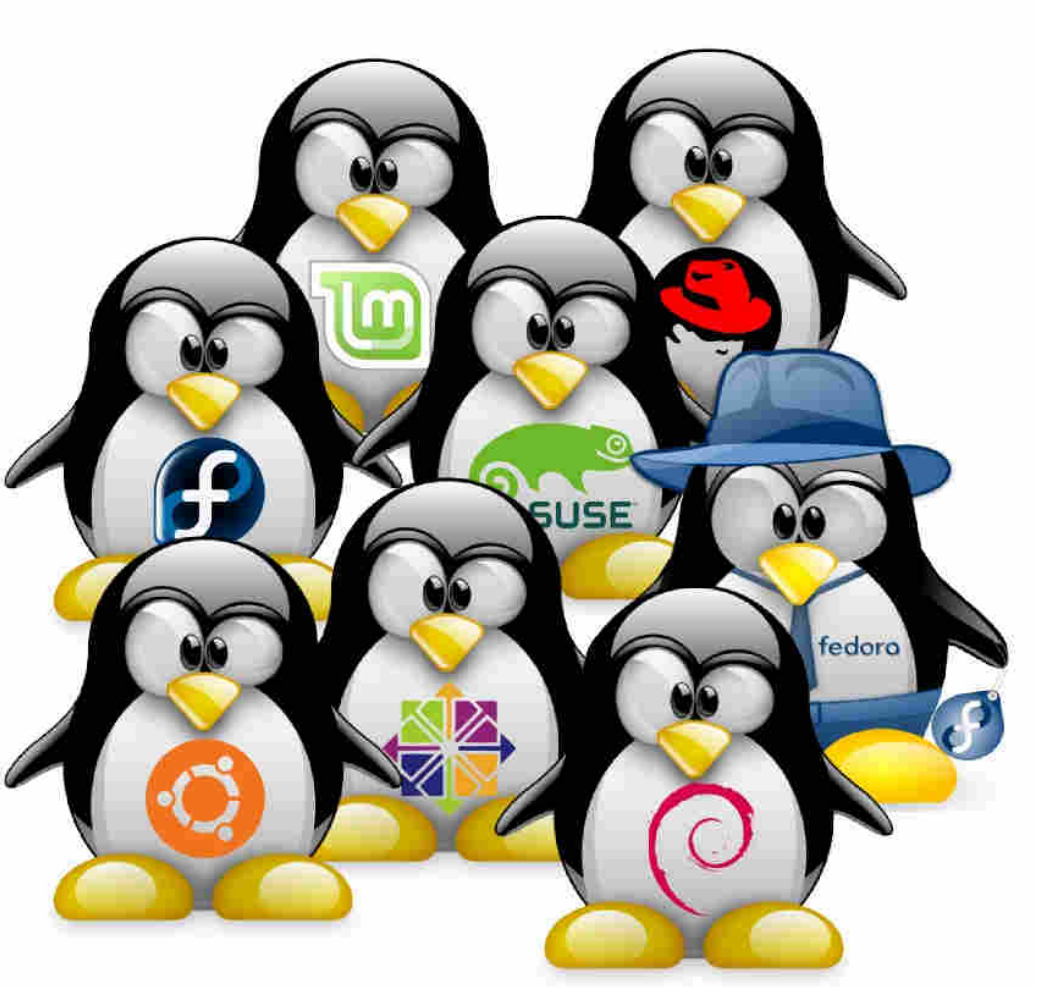
با فراخوانی تدریجی سیستم، کد آزاد بیشتری برای پهنه‌ی وسیعی از سخت‌افزارها منتشر میشد. این کدنویسان روی کامپیوترهای خودشان متوقف نشدند، هر قطعه‌ی سخت‌افزاری که میتوانستند پیدا کنند برای لینوکس مفید بود. در آن زمان آن دسته از مردم `nerds` یا `freaks` نامیده میشدند، اما این از زمانی که لیست سخت‌افزارهای پشتیبانی‌شده طولانی‌تر و طولانی‌تر میشد برایشان مهم نبود. بواسطه‌ی این افراد، لینوکس امروز تنها مناسب برای اجرا روی کامپیوترهای جدید نیست، بلکه همچنین سیستم انتخابی جهت سخت‌افزارهای قدیمی و کم‌نظیر هم هست به نحوی که اگر لینوکس وجود نداشت بلااستفاده بودند.

دو سال پس از پست لینوس، ۱۲۰۰۰ کاربر لینوکس وجود داشت. پروژه‌ای محبوب با همراهی علاقه‌مندان و رشد مداوم است، در حالی که در قلمرو استاندارد `POSIX` می‌ماند. همه‌ی ویژگی‌های یونیکس در چند سال آینده اضافه گشت و نتیجه آن سیستم عامل بالغ لینوکس امروزی است.

لینوکس یک `clone` کامل از یونیکس است، مناسب برای استفاده در ایستگاه‌های کاری و همچنین در سرورهای متوسط و سطح بالا میباشد.

۳.۱ چرا گنو/لینوکس

اگر جستجوی کوچکی در اینترنت با موضوع سیستم‌عامل لینوکس انجام دهید با مطالب زیادی درخصوص مقایسه‌ی سیستم‌های‌عامل، مزایا و معایب، موارد استفاده و ... روبرو میشوید. احتمالاً پس از مطالعه و بررسی چند مقاله سردرگم میشوید و هنوز سوال اصلی‌ای که در ذهن دارید این است که آیا گنو/لینوکس میتواند انتخاب خوبی باشد؟ در اینجا تلاش کرده‌ایم ویژگی‌های اصلی این سیستم‌عامل را بصورت خلاصه بیان کنیم:



۱. متنوع است.

مخالفان استدلال میکنند که تنوع لینوکس یک جور مشکل پراکندگی است، اما درحقیقت این یکی از

یزرگترین نقاط قوت آن محسوب میشود. کاربران انتخاب‌های بیشماری دارند. کسی ممکن است مینت یا اوبونتو را بخاطر تاکید بر قابل استفاده‌بودن دوست داشته باشد، شخص دیگری فدورا با ویژگی‌های متعدد سازمانی و امنیت بیشتر را ترجیح دهد، حتی انواعی که روی صنایع خاص متمرکز باشند هم وجود دارد. در دنیای لینوکس برای هر کسی چیزی وجود دارد.

۲. قابل سفارشی‌سازی است.

نه تنها میتوانید توزیع لینوکس خاصی را انتخاب کنید، بلکه یکی از مشخصه‌های لینوکس این است که قابلیت سفارشی‌سازی بالایی دارد. میزکار جدید اوبونتو یونیتی یا گنوم ۳ مینت را دوست ندارید؟ مسئله‌ای نیست، انتخاب‌های بسیار زیادی دارید و انتخاب شما به راحتی قابل نصب است. هیچ فروشنده‌ای در کار نیست که به شما دیکته کند که از کامپیوترتان چه شکلی استفاده کنید.

۳. متن باز است.

بخش اعظمی از انعطاف‌پذیری لینوکس از این واقعیت ناشی میشود که لینوکس متن باز است و به این معنی است که هیچ نهاد دیگری برای کنترل کد وجود ندارد. هر توسعه‌دهنده، هر کاربر میتواند کد را ببیند و به هر شکلی که مناسب است ویرایش کند.

۴. رایگان است.

لینوکس هیچ هزینه‌ای دربر ندارد. این واقعیت است، مگر اینکه انتخاب تجاری همراه با پرداخت هزینه‌ی پشتیبانی داشته باشید. اما لینوکس هنوز هم از حق ثبت اختراع و محدودیت‌های استفاده به دور است. رایگان و متن باز بودن مثل ضرب‌المثل شکر و تخم‌مرغ در کیک، ترکیب خوبی است.

۵. قابل اعتماد است.

این دلیلی است که چرا لینوکس در دنیای سرورها مثل یک دژ میماند. وقتی در لینوکس هستید لازم نیست ساعت‌های بهره‌وری از دست رفته بخاطر کرش یا خرابی را به یاد داشته باشید.

۶. سریع است.

لینوکس منابع سخت‌افزاری کمتری نسبت به سیستم‌های‌عامل دیگر نیاز دارد. حتی توزیع‌هایی برای سیستم‌های پایین هم طراحی شده‌اند. لینوکس حتی روی سخت‌افزارهای قدیمی هم سریع است.

۷. امن است.

سیستم‌عامل ویندوز طعمه‌ی اصلی ویروس‌ها و نرم‌افزارهای مخرب است اما هنوز هم به طور گسترده‌ای استفاده می‌شود. هیچ سیستم‌عاملی به طور کامل امن نیست، البته در دنیای لینوکس نرم‌افزار مخرب

به دلایل زیادی نادر هست. اگر این مسئله برای شما حائز اهمیت است، توزیع‌هایی با امنیت بیشتر در دسترس میباشد.

۸. به خوبی پشتیبانی میشود.

امروزه در کنار گزینه‌های پشتیبانی پولی، راه‌های بیشماری برای دریافت کمک رایگان از طریق انجمن‌های فعال کاربران و توسعه‌دهندگان برای اغلب توزیع‌ها وجود دارد.

۹. همواره در حال بهبود است. انجمن‌های توزیع‌های لینوکس به طور مداوم بواسطه‌ی ارائه‌ی ویژگی‌های جدید و رفع سریع آسیب‌پذیری‌ها خود را بهبود میبخشند. دیگر نیازی به انتظارهای چندماهه برای پچ‌ها و اصلاحیه‌ها نیست.

۱۰. سازگار است.

لینوکس نه تنها قصد دارد که با خواسته‌های کاربران هماهنگ باشد، بلکه تلاش میکند قابلیت همکاری بهتری نسبت به سیستم‌های‌عامل دیگر ارائه کند. اگر تاکنون در شرکت‌ها و نقاط دیگر دنیا با مردم همکاری کرده‌اید، بهترین سرمایه‌گذاری شما روی سیستم‌عاملی است که به جای موارد خاص، متعهد به پشتیبانی از استانداردهای بین‌المللی باشد[۱].

[۱] www.pcworld.com

۴.۱ کاربردهای لینوکس

سرور

لینوکس به شکل سنتی به عنوان یک سیستم عامل مناسب برای سرور شناخته شده است. این امر، ارث پدر معنوی لینوکس یعنی یونیکس است. بنا به آخرین گزارش‌های سایت نت‌کرافت، پنجاه درصد بهترین سرویس دهندگان وب از لینوکس بر روی سرورهای خود استفاده می‌کنند[۱].

این سیستم‌عامل همچنین نقش بسیار مهمی در سرورهای مخابراتی دارد و توزیع‌هایی با پشتیبانی تجاری همچون ردهت، درصد زیادی از سیستم عامل سرورها را به خود اختصاص داده‌اند.

کاربرد دیگر لینوکس در دنیای حرفه‌ای، مین‌فریم‌ها و سوپرکامپیوترها هستند. غول دنیای مین‌فریم یعنی آی‌بی‌ام اخیرا اعلام کرده که از سال ۲۰۰۹ به بعد، مین فریم‌هایش را با لینوکس عرضه خواهد کرد. همچنین طبق آخرین آمار سریعترین کامپیوترهای دنیا، از ۵۰۰ سوپرکامپیوتر برتر دنیا، ۸۸.۶٪ آن‌ها از لینوکس به عنوان سیستم‌عامل خود استفاده می‌کنند.

سیستم‌های درون‌ساخت

سیستم‌های لینوکس درون‌ساخت[۲] به سیستم‌هایی می‌گویند که لینوکس در آن‌ها به عنوان بخشی از یک ابزار خاص منظوره استفاده شده است. مثلا در یک موبایل، توستر، خودرو، قهوه‌ساز، ساعت یا غیره. تفاوت یک سیستم درون‌ساخت با یک توزیع معمولی لینوکس این است که در وضعیت درون‌ساخت، سخت افزار مورد استفاده کاملا مشخص است و در اکثر موارد هم منابع محدودی دارد. علاوه بر این، معمولا سیستم‌عامل برای اجرای یک برنامه از پیش آماده و ثابت استفاده می‌شود؛ در حالی که از یک توزیع معمولی انتظار می‌رود از سخت‌افزارهای متنوع پشتیبانی کند و بتواند طیف وسیعی از برنامه‌ها را اجرا کند.

در سیستم‌های درون‌ساخت هم لینوکس گزینه مناسبی است. اولا به خاطر قابلیت تغییر ساده، ماژولار، و بعد هم به خاطر هزینه‌های پایین. در حال حاضر لینوکس در تلفن‌های موبایل، انواعی از روترها و فایروال‌های شبکه (به‌ویژه در لینک‌سیس‌های سیسکو سیستمز) و ابزار دیگر استفاده می‌شود. با پیدایش سیستم‌عامل اندروید گوگل، استفاده از لینوکس درون‌ساخت در وسایل ارتباطی شتاب خیلی بیشتری گرفت.

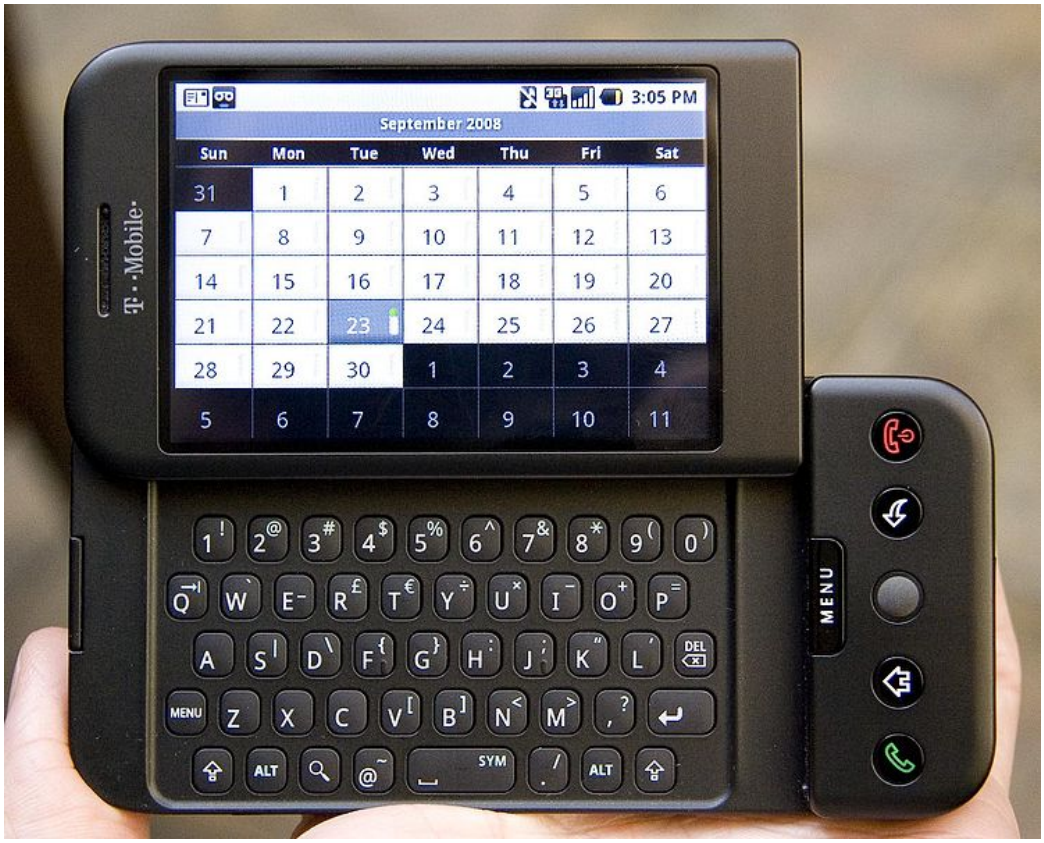
دسکتاپ

لینوکس چند سالی بیشتر نیست که به شکل جدی وارد دنیای دسکتاپ شده است. تقریبا ده سال پیش اگر می‌خواستید لینوکس را روی دسکتاپ یا لپ‌تاپ خود نصب کنید، دردسرهای خیلی زیادی داشتید اما حالا،

---

[۱] از پنجاه درصد باقی مانده سی درصد سهم فری‌بی‌اس‌دی و بیست درصد سهم مایکروسافت است.

[۲] Embedded



نمونه‌ای از یک گوشی اچ‌تی‌سی مجهز به لینوکس اندروید

همه چیز راحت تر است. در حال حاضر تقریبا تمام توزیع‌ها به شکل پیش‌فرض از محیط گرافیکی استفاده می‌کنند (مثال نقض جذاب، آرچ است) و به خاطر استفاده از کرنل‌های جدید، امکان شناسایی طیف بسیار وسیعی از سخت‌افزارها را دارند.

این روزها تقریبا نسخه جدید هر توزیعی را که داشته باشید، انتظار می‌رود بدون مشکل روی دسکتاپ یا لپ‌تاپ شما نصب شود و بدون دردسر به شما اجازه استفاده یا نصب طیف بسیار وسیعی از نرم‌افزارها را بدهد. از برنامه‌های آفیس گرفته تا بازی‌های سه بعدی، پخش کننده‌های موسیقی، چت، و ضبط کننده‌های تصاویر.

در حال حاضر در عمل تمام برنامه‌های عمومی موجود برای ویندوز و مک، نسخه‌های مشابه لینوکسی هم دارند. گاهی این برنامه، دقیقا نسخه مشابهی است که روی لینوکس کامپایل شده (مانند فایرفاکس) یا نسخه‌ی لینوکسی یک برنامه ویندوزی است (مانند اسکایپ) یا برنامه‌ای است که قابلیت کاری مشابهی با برنامه ویندوزی یا مکی دیگری دارد (مانند گیمپ که جایگزین فتوشاپ است).

البته نکته‌ای که نباید فراموش کنید این است که لینوکس ویندوز نیست. نباید انتظار داشته باشید که دقیقا هر چیزی روی ویندوز اتفاق می‌افتاد روی لینوکس هم اتفاق بیافتد. گاهی برنامه‌های تخصصی وجود دارند که مشابه لینوکسی ندارند (مانند یک برنامه‌ی خاص طراحی مدار) و گاهی شکل رابط کاربری یک برنامه کاملا با همکار ویندوزی‌اش فرق می‌کند (مثلا در مورد گیمپ).

همچنین ذات باز و آزاد لینوکس، به افراد اجازه داده تا توزیع‌های ویژه کاربردهای خاص را ایجاد کنند. مثلا توزیعی مثل پارسیکس با هدف پشتیبانی پیش فرض از زبان فارسی ساخته شده یا توزیعی مثل میت تی‌وی. به طور خاص برای تبدیل یک کامپیوتر به یک مدیاسنتر توسعه یافته است. این موضوع هم باعث شده درصدی از کاربران خانگی به سراغ لینوکس‌های خاص منظوره کشیده شوند.

کمک به جامعه‌ی بشری و ساختن یک دنیای بهتر

پروژه‌ی "یک لپ‌تاپ برای هر کودک[۱]" One Laptop per Child (OLPC) پروژه‌ای است با هدف بالابردن سطح آموزش در فقیرترین کشورهای جهان که توسط چند سازمان غیرانتفاعی دنبال می‌شود. این لپ‌تاپ‌های ارزان‌قیمت و کم‌قدرت (این موسسه اخیراً نوعی تبلت نیز در همین راستا ارائه کرده است) که XO نامیده میشوند مخصوص کودکان ساخته شده‌اند. سیستم عاملی که در این لپ‌تاپ‌ها استفاده می‌شود نوعی لینوکس مبتنی بر توزیع Fedora هست که از رابط کاربری Sugar استفاده میکند.

[۱] www.laptop.org

شاید خوشبین‌ترین افراد نیز تصور اینکه پروژه‌ی گنو/لینوکس و تلاش‌های استالمن، توروالدز و هزاران انسان دیگر روزی بدین شکل برای کمک به جامعه‌ی بشری و گسترش آموزش و پرورش در اینچنین ابعاد جهانی به خدمت گرفته شود، را نداشتند اما امروز این خواسته محقق گشته است. موسسه‌ی OLPC در وب‌سایت خود ماموریت این نهاد را اینگونه شرح می‌دهد: "آموزش شالوده‌ای است برای انسان کامل، توسعه‌ی اجتماعی، اقتصادی و دموکراتیک. با دسترسی به این نوع ابزار، کودکان در آموزش خود درگیر میشوند، یاد میگیرند، به اشتراک میگذارند و باهم خلق میکنند. آنها به یکدیگر، به جهان و به یک آینده‌ی روشن‌تر ملحق میشوند."

۵.۱ لینوکس ویندوز نیست

این چند وقت بحث زیادی در مورد لینوکس و ویندوز در گرفته. من اصولا این بحث را زیاد مربوط نمی‌دانم چون کامپیوتر و به تبع آن سیستم عامل یک **ابزار** است و تا وقتی ما **هدف** از استفاده را نشناسیم، انتخاب **ابزار** بی‌معنی است. من از لینوکس استفاده می‌کنم و برای اینکار دلایل زیادی دارم . در عین حال به نظرم در آینده مردم بیشتر و بیشتر از گنو/لینوکس استفاده خواهند کرد اما در عین حال معتقدم بدترین کاری که برای لینوکس می‌شود کرد این است که جوی بوجود بیاوریم که افراد بدون درک از اینکه هدفشان چیست، به این امید که همه مشکلاتشان حل شود آن را نصب کنند.

به هرحال... به عنوان مشارکت در این بحث‌ها، این مقاله مشهور "لینوکس ویندوز نیست[۱]" را اینجا ترجمه می کنم.

مشکل ۱: لینوکس دقیقا ویندوز نیست

شاید تعجب کنید ولی خیلی‌ها از این موضوع اظهار شکایت می‌کنند. این آدم‌ها از ویندوز به لینوکس می‌آیند تا یک سیستم‌عامل دقیقا مشابه ویندوز ولی آزاد داشته باشند. حتی در مواردی می‌بینیم که این حرف که لینوکس دقیقا مشابه ویندوز است را طرفداران افراطی لینوکس برای جلب ویندوزی‌ها می‌گویند.

دلیل آمدن افراد به لینوکس متنوع است اما شاید بشود همه آن‌ها را در یک دلیل خلاصه کرد: داشتن سیستم‌عاملی بهتر. اما بهتر بودن را چگونه باید سنجید؟ فاکتورهای بسیاری در بهتر بودن و بدتر بودن یک سیستم عامل مهمند. مثلا هزینه، حق انتخاب، کارایی، ایمنی و خیلی چیزهای دیگر. اما به هرحال شکی نیست که هر کسی که از ویندوز به لینوکس می‌آید، این کار را می‌کند تا به چیز بهتری برسد.

اما یک مشکل هست. از نظر منطقی غیر ممکن است چیزی از چیز دیگر بهتر باشد بدون اینکه با آن تفاوت داشته باشد. اگر کسی لینوکس را امتحان کند و انتظار داشته باشد که با چیزی بهتر از ویندوز رو برو شود، باید هم انتظار داشته باشد که با چیزی متفاوت روبرو شود. خیلی‌ها این واقعیت را نادیده می‌گیرند و وجود هر تفاوت را به عنوان نقص در لینوکس استنباط می‌کنند.

---

[۱] Linux is NOT Windows

به عنوان مثال، به مساله به روز رسانی درایورها دقت کن. در ویندوز آدم‌ها برای به روز رسانی درایورها به وب سایت تولید کننده می‌روند و نسخه جدید را دانلود و اجرا می‌کنند. در لینوکس این کار با به روز رسانی کرنل انجام می‌شود.

در واقع در لینوکس دانلود یک کرنل و نصب آن کل درایورها را به روزرسانی می‌کند در حالی که در ویندوز باید به کلی سایت سر بزنید و هر به روز رسانی را به شکل جداگانه انجام دهید. این دو روش کاملا با هم فرق دارند ولی خیلی‌ها شکایت می‌کنند که چرا درایورها در لینوکس مثل کرنل به روز رسانی نمی‌شوند.

یا مثلا به فایرفاکس نگاه کنید که این روزها مشهورترین نمونه موفقیت نرم‌افزارهای بازمتن است. این نرم‌افزار مشهور شد چون از اکسپلورر بهتر بود. اگر قرار بود شبیه آن باشد هیچ وقت نمی‌توانست جایش را بگیرد. فایرفاکس از همان اول از برگه‌ها (تب‌ها) پشتیبانی می‌کرد، بوک‌مارک زنده داشت، امکان جستجو داشت، تصاویر PNG را نشان می‌داد، می‌توانست جلوی تبلیغات ناخواسته را بگیرد و ... قابلیت جستجو در نوار ابزاری در قسمت پایین نمایان شده و به دنبال همانندهایی که تایپ کرده‌اید میگردد، وقتی نتیجه‌ای به دست نیاید قرمزرنگ می‌شود. در عوض IE تب که ندارد، از RSS سر در نمی‌آورد، خودش جای جستجو در گوگل و یاهو و ... ندارد و اگر بخواهید چیزی را جستجو کنید باید یک پنجره باز کنید و بعد OK را بزنید و اگر چیزی پیدا نشد ایراد Not Found را ببینید. این نمونه‌ای است از اینکه چرا یک نرم‌افزار بهتر، با تفاوتی که دارد پیروز می‌شود. اگر فایرفاکس، کپی آی.ای. بود هیچ امیدی نداشت. دقیقا همانطور که اگر قرار بود لینوکس عین ویندوز باشد، به هیچ دردی نمی‌خورد.

پس راه حل ایراد اول این است که به یاد بسپاریم که لینوکس با ویندوز فرق دارد. درست است که این دو شبیه هم هستند و تقریبا همانطور که با ویندوز کار می‌کردید می‌توانید با لینوکس هم کار کنید ولی تلاش لینوکس به هیچ وجه این نیست که شبیه ویندوز باشد. لینوکس یک ویندوز بهتر نیست. اگر به لینوکس آمده‌اید خوش آمده‌اید، اینجا چیزها با ویندوز فرق دارند چون فقط با همین کار است که می‌توانند بهتر باشند.

مشکل ۲: لینوکس خیلی با ویندوز فرق دارد

حالا با کسانی طرفیم که پذیرفته‌اند لینوکس و ویندوز دقیقا یک چیز نیستند، اما تفاوتی که می‌بینند بیشتر از آن است که انتظارش را دارند. شاید بهترین مثال، حق انتخاب بسیار وسیع در لینوکس باشد.

اگر کسی به شما بگوید که یک کامپیوتر ویندوز XP خریده، کاملا می‌دانید که بعد از زدن به برق با چه شکلی و چه نرم‌افزارهایی روبرو خواهد شد: آن پس‌زمینه تکراری، وردپد، اکسپلورر، آوت لوک اکسپرس و ... . اما همچین چیزی در مورد لینوکس غیرممکن است. حتی کل رابط گرافیکی ممکن است متفاوت باشد. از گنوم و کی.دی.ای. گرفته تا فلاکس‌باکس و اوپن‌باکس از اوپرا و فایرفاکس گرفته تا کانکوئرور و گالئون و دیلو و .. .

خیلی پیش می‌آید که کاربر ویندوز که به اینهمه تنوع عادت ندارد گیج یا حتی وحشت‌زده شود. آیا

واقعا این میزان از تنوع لازم است؟ مگر سیستم عامل چیزی بیشتر از یک برنامه است که به ما اجازه می‌دهد برنامه‌های دیگر را اجرا کنیم؟ برای جواب به دنیای اتومبیل‌ها نگاه کنید. مگر ماشین چیزی بیشتر از وسیله‌ای است که به ما اجازه می‌دهد از نقطه الف به نقطه ب برویم؟ شاید با خودتان بگویید که تفاوت خودروها فقط در رنگ و قیافه است وگرنه همه یک فرمان دارند و یک گاز و چند دنده و ... . موتورسیکلت چی؟ تراکتور؟ هواپیما؟ (: ماشین کورسی چی؟

رفتن از یک ویندوز به یک ویندوز دیگر مثل عوض کردن یک ماشین معمول به یک ماشین معمول دیگر است (: اما تغییر از ویندوز به لینوکس مثل عوض کردن خودرو با موتورسیکلت، ماشین کورسی و ... است. هنوز هم از برنامه‌ای برای اجرای برنامه‌های دیگر استفاده می‌کنید اما دیدگاه کاملا عوض شده است. مثلا در خودروی ویندوز لازم بود حتما آنتی ویروس نصب کنید تا از شر مزاحمان در امان بمانید اما مزاحمان جاده (چه ویروس‌ها، چه هکرها و چه تبلیغات) از نظر فنی امکان ورود به کاروان لینوکسی شما را ندارند.

یا از یک زاویه دیگر:

خودروی لینوکس از اول برای چندین مسافر طراحی شده در حالی که موتورسیکلت ویندوز تک سرنشین طراحی شده. از اول برنامه این بوده که هر کسی روی موتورسیکلت ویندوز نشست، همه کاره این وسیله نقلیه باشد ولی در اتوموبیل لینوکس کسی خودرو را کنترل می کند که پشت فرمان نشسته و بقیه می‌توانند سوار و پیاده شوند ولی الزاما پشت فرمان ننشینند.

خیلی چیزها هستند که استفاده از موتور یا ماشین، در آن‌ها **تفاوتی ایجاد نمی‌کند** . مثلا استفاده از بنزین، جاده، قوانین راهنمایی و رانندگی، علامت دادن قبل از پیچیدن و محدودیت‌های سرعت. اما چیزهایی هم هستند که **تفاوت دارند** . در ماشین لازم نیست کلاه ایمنی بگذارید، موتورسوارها کمربند ایمنی ندارند، ماشین‌ها با چرخاندن فرمان دایره‌ای کنترل می‌شوند و موتورها با چپ و راست کردن یک دسته و وزن بدن و ...

مشکل ۳: شوک فرهنگی

زیرمشکل ۳ الف: فرهنگی که از قبل وجود دارد[1]

کاربران ویندوز - حداقل در صورتی که نرم افزار را ندزدیده باشند - یک رابطه مشتری و فروشنده با ارائه دهنده سرویس دارند. آن‌ها برای نرم افزار افزار، گارانتی، پشتیبانی و هر چیز دیگر پول می‌دهند. آن‌ها انتظار دارند که نرم افزاری که خریده اند تا حد قابل قبولی درست کار کند و نسبت به آن احساس حق می‌کنند: آن‌ها پول داده‌اند و حق دارند که آن را کامل و با پشتیبانی دائمی تحویل بگیرند. در عین حال طرف حساب آن‌ها شرکت ها هستند نه افراد.

[1] این قسمت چندان به ایران مربوط نیست. من و شما در اکثر موارد برای نرم افزار پول نمی‌دهیم و حتی اگر هم بدهیم، از پشتیبانی نمی‌توانیم استفاده کنیم.

کاربران لینوکس بر خلاف حالت قبل، تشکیل یک جامعه را می‌دهند. آن‌ها برای نرم افزار و خدمات پول نمی‌دهند. نرم افزار را به رایگان دریافت می‌کنند و از طریق اینترنت از جامعه یا نویسندگان کمک می‌گیرند. آن‌ها با آدم‌ها طرف هستند نه شرکت‌ها.

اگر کاربر ویندوز با همان عقاید و رفتارهایی که سابقا داشته، به دنیای لینوکس بیاید، به چیزی که می‌خواهد نخواهد رسید.

این تضاد را شدیدا در فروم‌ها می بینید: کاربر اول که تازه به دنیای لینوکس آمده یک سوال مطرح می‌کند. وقتی جوابی که می‌خواهد را نمی‌گیرد، شروع به انتقاد می‌کند و کمک بیشتری می‌طلبد. این همان چیزی است که در دنیای تجاری اتفاق می‌افتد. مساله این است که اینجا کسی برای کمک، حقوق نمی‌گیرد. اینجا آدم‌های داوطلبی هستند که می‌خواهند به بقیه کمک کنند چون اینکار برایشان جذاب است.

دقیقا به دلیل بالا، در دنیای ویندوز شرکت‌ها تا نرم‌افزارشان به سطح کاملا قابل قبولی از پایداری نرسد، آن را منتشر نمی‌کنند چون توان پشتیبانی تجاری یک نرم‌افزار تکمیل نشده را ندارند. اما در دنیای لینوکس، هر نرم افزار از اولین نسخه‌هایش به تست عمومی گذاشته می‌شود. با اینکار کسانی که به آن علاقمند هستند می‌تواند در اولین لحظات نوشته شدن از آن استفاده کنند و حتی برنامه نویسان می‌توانند بخشی از نوشتن بقیه آن را بر عهده بگیرند. در این حالت کل جامعه درک می‌کند که این نرم‌افزار هنوز به مرحله نهایی نرسیده است.

در این جهان، یک تازه وارد ویندوزی ممکن است حس کند برنامه‌ای که استفاده می‌کند قابلیت‌های فنی لازم را ندارد و شروع به غر زدن کند. تصور کنید جوابی مثل این بشنود که «اگر نمی خواهی، همانقدر که پول داده‌ای را پس بگیر و برو!» و به این نتیجه برسید که چه حسی به این تازه وارد دست خواهد داد. برای فائق آمدن به این مشکل به یاد داشته باشید که نرم افزار را رایگان دریافت کرده‌اید و به کسی هم برای پشتیبانی پول نداده‌اید.

زیرمشکل ۳ ب : جدید در مقابل قدیم

> لینوکس به عنوان یک سرگرمی یک هکر ابداع شد. بعد هکرهای دیگر جذب آن شدند و حتی تا یکی دو سال بعد هم هیچ کس به جز گیک‌های حرفه‌ای نمی‌توانستند یک کامپیوتر لینوکسی را راه اندازی کنند. لینوکس در ابتدا یک نرم افزارها «از گیک، برای گیک» بود و حتی امروز هم اکثریت کاربران لینوکس، کسانی هستند که با افتخار خودشان را گیک می‌نامند.

این یک مزیت ایجاد می‌کند: اگر با سخت افزار یا نرم افزار مشکلی داشته باشید، کلی گیک در دنیا هستند که حاضرند روی مشکل شما کار کنند. اما لینوکس از آن روزها پیشرفت زیادی کرده. این روزها توزیع‌هایی (در واقع انواع مختلفی از گنو لینوکس) وجود دارند که برای نصب نیاز به هیچ سواد فنی ندارند. توزیع هایی هستند

که به شکل دیسک زنده بوت می‌شوند و بدون هیچ دخالتی از جانب شما، همه سخت افزارها را می‌شناسند (چیزی که در ویندوز غیر ممکن است). این روزها بعضی از غیرگیک‌ها هم دارند به لینوکس به عنوان یک سیستم عامل امن، بدون ویروس، سریع و آزاد علاقمند می‌شوند.

این دو شاخه مختلف لینوکس، گاهی با هم بحث‌شان می‌شود یا با هم تضاد پیدا می‌کنند. چیزی که باید به یاد داشته باشید این است که دو طرف این بحث، با وجود داشتن دیدگاه‌های متفاوت می‌توانند با هم کنار بیایند. هیچ یک از این دو دیدگاه، بد ذات نیستند و نمی‌خواهند طرف مقابل را تخریب کنند. بگذارید دقیق‌تر نگاه کنیم.

در طرف اول، گیک‌های حرفه‌ای هستند که فکر می‌کنند هر کس از لینوکس استفاده می‌کند باید یک گیک باشد. گیک بودن یعنی داشتن دانش فنی عمیق یا علاقه و آمادگی به کسب آن. این سبک زندگی گاهی منجر به خشونت در کلام نسبت به کسانی که نمی‌خواهند این دانش را کسب کنند می‌شود.

در طرف دوم، آدم‌های غیرفنی‌ای هستند که بعد از یک عمر استفاده از سیستم‌های تجاری دیگر، به دنیای لینوکس آمده‌اند. این آدم‌ها به نرم‌افزارهایی عادت دارند که هر کسی می‌تواند بعد از یک نصب ساده، شروع به استفاده از آن کند.

مشکل اینجا حاد می‌شود که گروه یک دوست دارد بتواند سیستم عامل را تا عمیق‌ترین لایه تنظیم و بازسازی کند در حالی که برای گروه دوم، سیستم عامل چیزی است که باید بدون هیچ دخالت و دردسری، نصب و اجرا شود.

بذارید با لگو مثال بزنم. همان مکعب‌های کوچکی که با وصل شدن به هم می‌توانند هر چیزی بسازند.

این صحنه را تصور کنید:

تازه وارد: من دنبال یک ماشین اسباب بازی جدید بودم و دیدم همه در مورد باحال بودن لگو حرف می‌زنند. پس رفتم و یک ماشین لگویی خریدم اما وقتی به خانه رسیدم کلی مکعب و چرخ و این جور چیزها در جعبه بود. پس ماشین من کجاست؟

قدیمی: خب باید ماشین‌ات را از همان مکعب‌ها درست کنی دیگر. این دقیقا چیزی است که قرار است لگو باشد.

تازه وارد: چی؟ من چه می‌دانم چطور باید ماشین ساخت! من که مکانیک نیستم. از کجا قرار است ماشین ساختن بلد باشم؟

قدیمی: حتما در جعبه یک راهنما هم هست. راهنما دقیقا می‌گوید که ماشین چطور باید ساخته شود. لازم نیست چیزی را از قبل بدانی. فقط باید راهنماها را دنبال کنی.

تازه وارد: آه درست است. راهنما اینجاست. ولی درست کردنش ساعت‌ها طول می‌کشد! خب چرا یک ماشین کامل در جعبه نگذاشته‌اند؟ مگر من باید ماشین را درست کنم؟

قدیمی: خب تو حتما لازم نیست بخواهی ماشین درست کنی. اما کسانی که دوست دارند، لگو می‌خرند. خوبیش این است که می‌توانی از لگو هر چیز دیگری هم درست کنی. کل ایده همین است.

تازه وارد: ولی من که نمی‌فهمم چرا باید ماشین درست کنم! خب چرا مثل آدم یک ماشین کامل در جعبه نبود تا اگر کسی خواست بازش کند و دوباره خودش بسازدش... به هرحال... حالا که بعد از چند ساعت کار ماشین را ساخته‌ام جالب شده ولی گاهی بعضی از قسمت‌ها جدا می‌شود. می‌شود آن‌ها را با چسب بچسبانم؟

قدیمی: خب می‌شود.. ولی دیگر لگو نخواهد بود. این سیستم درست شده تا قطعاتش قابل جدا کردن و وصل کردن دوباره باشد.

تازه وارد: ولی من دوست ندارم قعطاتش جدا شود! من فقط می‌خواهم یک ماشین اسباب بازی داشته باشم

قدیمی: **خب لعنتی! پس چرا لگو خریده‌ای؟!!**

همه می‌دانند که لگو به درد کسانی که می‌خواهند فقط یک ماشین اسباب بازی داشته باشند نمی‌خورد. کسی که لگو می‌خرد، دوست دارد با آن بازی کند و چیزهایی که می‌خواهد را بسازد. اگر علاقمند به ساختن چیزهایی که در ذهن‌تان هست نیستید، لگو به درد شما نمی‌خورد. این کاملا واضح است.

تا جایی که به کاربر قدیمی مربوط می‌شود، لینوکس هنوز همان لگوی دست داشتنی قدیمی است: یک سیستم بازمتن و آزاد که کاملا قابل تنظیم شدن توسط کاربر است. این بهترین ویژگی لینوکس است. اگر نمی‌خواهید با اجزای تشکیل دهنده سیستم عامل خودتان ور بروید، چه دلیلی دارد سراغ لینوکس بیایید؟

اما اخیرا تغییرات زیادی به نفع کاربر تازه وارد انجام شده و لینوکس بیشتر و بیشتر با نیازهای غیرهکرها سازگار شده است. البته گفتگوهایی مثل متن بالا هنوز هم تکرار می‌شوند. کاربران تازه وارد از اینکه برای استفاده از چیزی باید راهنما بخوانند، تعجب می‌کنند ولی به هرحال از نظر آن‌ها، چیزی که دریافت می‌کنند یک لگوی از پیش ساخته است که اگر کسی بخواهد می‌تواند آن را از هم باز کند و اگر نخواهد می‌تواند تا ابد از همان ماشین اسباب بازی که در کارخانه سر هم شده، لذت ببرد. البته هنوز این شکایت را هم می‌بینیم که انتخاب بین توزیع‌های مختلف زیاد است یا انتخاب‌های زیادی هستند برای نصب هر نرم‌افزار یا بعضی سخت‌افزارها سازگاری لازم را ندارند. این مثل این است که کسی شکایت کند که لگو انواع مختلفی اسباب بازی می‌سازد و ناراحت باشد از اینکه لگو سعی می‌کند تمام اسباب بازی‌هایش را قابل باز شدن و دوباره ساختن، ارائه دهد.

پس برای جلوگیری از زیرمشکل ۳ ب باید به یاد داشته باشید که این روزها لینوکس دقیقا آن چیزی که قدیم‌ها بود نیست. اما کماکان حجم زیادی از جامعه پشت لینوکس - از جمله هکرها و توسعه دهندگان آن - آن را به همان دلایل قدیمی دوست دارند.

مشکل ۴: طراحی شده برای طراح

در صنعتی مثل صنعت اتوموبیل، تقریبا محال است ببینید همان کسی که موتور خودرو را طراحی کرده، بدنه را هم رنگ‌آمیزی کند. همین مساله در دنیای نرم‌افزار هم صادق است؛ در اکثر موارد کسی که اصل برنامه را می‌نویسد با کسی که رابط گرافیکی را طراحی می‌کند متفاوت است.

اما در دنیای لینوکس، پروژه‌ها معمولا توسط یک نفر شروع می‌شوند و به عنوان یک سرگرمی (: در ابتدای کار یک نفر همه کارها را می‌کند و به همین دلیل او بیشتر دنبال نوشتن برنامه‌های حرفه‌ای است تا درست کردن رابط‌های کاربر پسند : کاربر اول این برنامه خود برنامه‌نویس است و همه چیز در مورد برنامه را می‌داند. سادگی استفاده برای او اصلا مهم نیست. مثلا `vi` را در نظر بگیرید. این برنامه دقیقا برای کسی طراحی شده که کار با آن را بلد است. یک کاربر تازه وارد ممکن است برای خارج شدن از آن، مجبور شود کامپیوترش را بوت کند!

یک تفاوت مهم دیگر هم بین نرم‌افزارهای آزاد و بازمتن (FOSS) و نرم‌افزارهای تجاری هست: برنامه بازمتن معمولا به خاطر استفاده برنامه‌نویس توسعه پیدا می‌کند در حالی که برنامه تجاری برای استفاده شدن توسط مشتری نوشته می‌شود. این مساله تیغ دو لبه است: از یک طرف کاربر با برنامه‌ای مواجه است که استفاده از آن چندان راحت نیست ولی در مقابل کاربر نهایی مطمئن است که این برنامه برای استفاده خود نویسنده طراحی شده و در نتیجه او تمام تلاش خود برای بالا بردن کیفیت برنامه را کرده است. در عین حال در این حالت کاربرنهایی دقیقا می‌داند که نویسنده برنامه درک می‌کند کاربرنهایی چه می‌خواهد چون خودش هم یکی از کاربران برنامه خواهد بود؛ درست برعکس یک برنامه تجاری. بازهم به `vi` نگاه کنید؛ رابط کاربری آن بسیار برای یک کاربر تازه کار سخت است اما در عوض آنقدر قوی است که هنوز هم به عنوان اصلی‌ترین ویرایشگر متن، استفاده می‌شود.

پس رابط‌های کاربری لینوکس معمولا برای کاربر تازه کار کمی عجیب هستند و چیزی مثل `vi` اصولا برای تازه کاری که می‌خواهد چند تغییر در یک فایل ایجاد کند، اصولا چیز مناسبی نیست. در عین حال اگر در حال استفاده از نسخه‌های اولیه و در حال توسعه یک نرم‌افزار باشید، احتمالا رابط کاربری خوب و راحت و زیبا فقط در بخش ToDo ی برنامه یافت خواهند شد. اولویت اول یک برنامه نویس، هسته اصلی برنامه است نه رابط گرافیکی آن. هیچ برنامه نویس بازمتنی اول یک رابط گرافیکی مکش مرگ من طراحی نمی‌کند تا بعدا سر فرصت سراغ نوشتن کارکرد اصلی برود. در این دنیا، اول کارکرد اصلی برنامه نوشته می‌شود و بعد قدم به قدم رابط گرافیکی بهبود می‌یابد.

پس برای جلوگیری از مشکل ۴: به سراغ برنامه‌هایی بروید که برای راحتی کاربر طراحی شده‌اند یا بپذیرید که در این دنیا روند یادگیری استفاده از برنامه‌ها ممکن است کندتر باشد. ایراد گرفتن از این که چرا استفاده از `vi` راحت نیست، باعث خواهد شد همه باور کنند که شما اصولا نکته را نگرفته اید (:

مشکل ۵: افسانه کاربرپسندی

یک مساله جدی. عبارت کاربرپسند یا User Friendly آنقدر مهم است که یکی از سایت‌های کاریکاتور بسیار مشهور هم اسمش را همین گذاشته (در ایران فیلتر است، چون در مورد برنامه نویسی است). اما این عبارت، عبارت ناجوری است.

ایده اصلی خوب است: نرم‌افزار باید بر اساس نیازهای کاربر نوشته شود. اما نمی‌شود به این مساله نگاهی تک بعدی داشت.

اگر در تمام طول زندگی مشغول تحلیل فایل‌های متنی باشید، نرم‌افزار مورد پسند شما یک نرم‌افزار سریع و قدرتمند خواهد بود که به شما اجازه بدهد بیشترین حجم کار را در کمترین وقت انجام دهید. شما از داشتن چند کلید میانبر و عدم نیاز به استفاده از ماوس برای کنترل برنامه، لذت خواهید برد.

اما اگر خیلی کم با فایل‌های متنی کار داشته باشید و فقط گاه گاهی بخواهید یک نامه تایپ کنید، بهتر است نیازی به یادگیری هیچ کلید میانبری نداشته باشید. در این حالت شما نیازمند منوهای خوب و شکلک/آیکن‌های واضح خواهید بود. نوارابزارها هم در این حالت می‌توانند حسابی مفید باشند.

شکی نیست که نرم‌افزاری که بر اساس نیازهای کاربر اول طراحی شده باشد، برای دومی مفید نیست. و البته برعکس. حالا به چه نرم‌افزار ویرایش متنی باید **کاربرپسند** بگوییم؟

جواب ساده: کاربرپسندی، لغت ناواضحی است که سعی می‌کند یک مفهوم پیچیده را ساده جا بزند.

کاربرپسند واقعا به چه معنا است؟ در واقع کاربرپسندی را باید به این معنی دانست: نرم‌افزاری که یک کاربر، بدون داشتن تجربه قبلی بتواند در حد معقولی از آن استفاده کند . توجه کنید که بنا بر این تعریف، برنامه‌هایی که از منوهای بد ولی آشنا برای شما استفاده کنند، کاربرپسند به نظر خواهند رسید.

زیرمشکل ۵ الف: آشنا، مورد پسند است

واقعیت این است که این روزها در اکثر ویرایشگرهای متن آشنا و کاربرپسند شما می‌توانید با Ctrl+X و Ctrl-V متون را Cut و Paste کنید. این دو کلید واقعا بی‌ربط به کاری که می‌کنند هستند اما چون آشنا هستند، کاربرپسند به نظر می‌رسند.

حالا اگر کسی سراغ vi بیاید و ببیند که d برای cut استفاده می‌شود و p برای paste ، این به نظرش ناکاربرپسند خواهد رسید چون به آن عادت ندارد.

حالا کدام مفیدتر است؟ vi (: با داشتن Ctrl+X ، چطور می‌توانید با کیبرد یک کلمه را cut کنید؟ اول باید به اول کلمه بروید و بعد Ctrl+Shift+Right را بزنید و بعد از اینکه انتخاب شد، Ctrl+X را فشار دهید.

در vi ؟ dw یا در اصل delete word.

حالا اگر بخواهید **پنج** کلمه را `Ctrl+X` کنید چه؟ به اول کلمه بروید و بعد `Ctrl+Shift+Right` را فشار دهید و بعد

```
Ctrl+Shift+Right
Ctrl+Shift+Right
Ctrl+Shift+Right
Ctrl+Shift+Right
Ctrl+X
```

و در `vi` ؟ `d5w` که در اصل همان `Delete 5 words` است.

می‌بینید که روش `vi` کاملا سریعتر و منطقی‌تر است. `X` و `V` هیچ ربطی به `cut` و `paste` ندارند اما `d` و `w` حروف اول `delete` و `word` هستند. اما... اما ما به `x` و `v` عادت کرده‌ایم و `vi` به نظرمان غیردوستانه می‌آید و اگر ادیتوری ببینید که شبیه ویندوز باشد، به نظرتان دوستانه خواهد آمد. یادتان هست؟ مشکل شماره ۱ این بود که لینوکس عین ویندوز نیست؛ دقیقا به همین خاطر برای یک تازه وارد لینوکس در اوایل غیردوستانه‌تر می‌آید.

برای غلبه بر مشکل ۵الف، باید به یاد داشته باشید که «کاربر پسند» به معنی «چیزی که کاربر به آن عادت دارد» نیست.

زیرمشکل ۵ ب: کاربرپسند، غیرکارا است

این یک واقعیت دوست نداشتنی است. متاسفانه هر چقدر راه دستیابی به کارکرد مورد نظر دورتر باشد، برنامه کاربرپسندتر به نظر خواهد رسید. چرا؟ چون اصولا کاربرپسندی، با اضافه کردن نشانه‌های تصویری به قابلیت اصلی اضافه می‌شود و هر چقدر این نشانه‌ها بیشتر باشند، برنامه کاربرپسندتر می‌شود. فرض کنید یک تازه وارد کامل به دنیای دیجیتال، پشت یک ویرایشگر متن ویزیویگ بنشیند (ویزیویگ یعنی برنامه‌ای که دقیقا چیزی را پرینت خواهد گرفت که شما روی صفحه می‌بینید، مثل برنامه ورد مایکروسافت یا رایتر اوپن آفیس) و بخواهد بخشی از یک متن را سیاه‌تر / بولد کند. احتمالا:

۱. باید حدس بزند که فشردن `Ctrl+B` متن را سیاه‌تر می‌کند، که بعید است.

۲. باید دنبال نشانه‌ها بگردد. احتمالا از منوی `Edit` شروع می‌کند

۳. و وقتی موفق نمی‌شود به سراغ منوی `Format` خواهد رفت

۴. و از آنجا `Font` را انتخاب خواهد کرد

۵. و به گزینه `Bold` خواهد رسید!

دفعه بعد که ویرایشگر متن تان را باز کردید، سعی کنید همه کارها را از طریق منوها انجام دهید. نه از میانبرها استفاده کنید و نه از آیکون‌های روی نوارابزار. وقتی با سرعت لاک پشت پیش رفتید کشف خواهید کرد که چرا کاربرپسند بودن، غیرکارامد است.

در این دیدگاه، «کاربر پسند بودن» مثل گذاشتن چرخ‌های تمرینی در کنار چرخ‌های اصلی دوچرخه است. با چنین دوچرخه‌ای هر کسی با هر سطحی از توانایی می تواند سوار دوچرخه شود. این چرخ‌ها برای یک نوآموز فوق العاده اند اما هیچ کسی که دوچرخه سواری بلد باشد به آن‌ها علاقه‌ای نشان نمی‌دهد و هیچ کس هم مدعی نمی‌شود که همه دوچرخه‌ها را باید به آن‌ها مجهز کرد.

اکثر برنامه‌های لینوکس برای کسانی که هنوز نیازمند چرخ‌های آموزشی هستند، نوشته نشده‌اند. تصور برنامه‌ها این است که شما علاقه‌مند هستید پیشرفت کنید. هیچ کس برای همیشه نوآموز نمی‌ماند و می‌دانید که اگر چیزی یاد بگیرید، برای همیشه آن را بلد خواهید بود. این استدلال به اینجا می‌رسد که اکثریت آدم‌ها، در وضعیت صفر نیستند و کمتر کسی است که در دنیای کامپیوتر نیازمند چرخ‌های آموزشی باشد.

شاید این استدلال را علیه این بشنوید: به هرحال ورد مایکروسافت هم همه این منوها را دارد ولی آیکون‌های نوارابزار و کلیدهای میانبر را هم دارد. در واقع این نرم‌افزار خوبی‌های هر دو دنیا را یکجا جمع کرده؛ کاربرپسند **و** کارا.

اما این دیدگاه را هم اضافه کنید: اول اینکه منوها و نوارابزار و همه این چیزها نیازمند برنامه‌نویسی هستند. برنامه‌نویسان لینوکس برای کارشان حقوق نمی‌گیرند پس سعی می‌کنند کدی که به آن علاقه ندارند را ننویسند. دوم اینکه اصولا کاربران خیلی حرفه‌ای از نرم‌افزارهایی مثل ورد مایکروسافت استفاده نمی‌کنند. آیا تا به حال برنامه‌نویسی فوق حرفه‌ای را دیده‌اید که از مایکروسافت ورد استفاده کند؟ نه. آن‌ها از `emacs` و `vi` استفاده می‌کنند. چرا اینطور است؟ اول اینکه بعضی از کارایی‌های «کاربرپسند» مجبورند قدرت را پایین بیاورند: برای مثال `cut` و `paste` که در بالا زدیم. مساله دوم این است که اکثر قابلیت‌های نرم‌افزاری مثل ورد در منوهایی هستند که در هر صورت مجبور به استفاده از آن هستید. شما فقط توانایی‌های عمومی را در

میانبرها و تولبارها پیدا می‌کنید و هنوز باید برای کاربردهای حرفه‌ای که معمولا حرفه‌ای ها به آن نیاز دارند، به سراغ منوهای پر دردسر بروید. راستی این را هم در نظر داشته باشید که چرخ‌های آموزشی، معمولا بخش اصلی نرم‌افزارهای لینوکس نیستند ولی در بسیاری از موارد، در صورت نیاز می‌توانید آن‌ها را نصب کنید.

مثلا برنامه پخش کننده `mplayer` را در نظر بگیرید. برای پخش یک فایل باید `mplayer filename` را در ترمینال وارد کنید. در حین پخش، کلیدهای چپ و راست و صفحه بالا و پایین، می‌توانند پخش را سریع به جلو یا عقب بروند. این چیزی نیست که معمولا به آن عادت دارید. پس بهتر است از `gmplayer filename` استفاده کنید تا به همان رابط گرافیکی همیشگی که معمولا علاقمند به دیدنش هستید، برسید.

یا بگذارید سراغ تبدیل یک `CD` صوتی به `MP3` (یا بهتر از آن `Ogg`) بروید: در خط فرمان باید از برنامه `cdparanoia` استفاده کنید تا فایل‌های دیسک را بخوانید. بعد باید از یک کد کننده استفاده کنید که (حداقل به نظر من) کار پچیده‌ای است. پس اگر من باشم چیزی مثل `Grip` را نصب می‌کنم تا با یک رابط گرافیکی دوبرنامه قبلی را در پشت صحنه اجرا کند و تبدیل سی.دی. به فایل صوتی را به کاری ساده تبدیل کند.

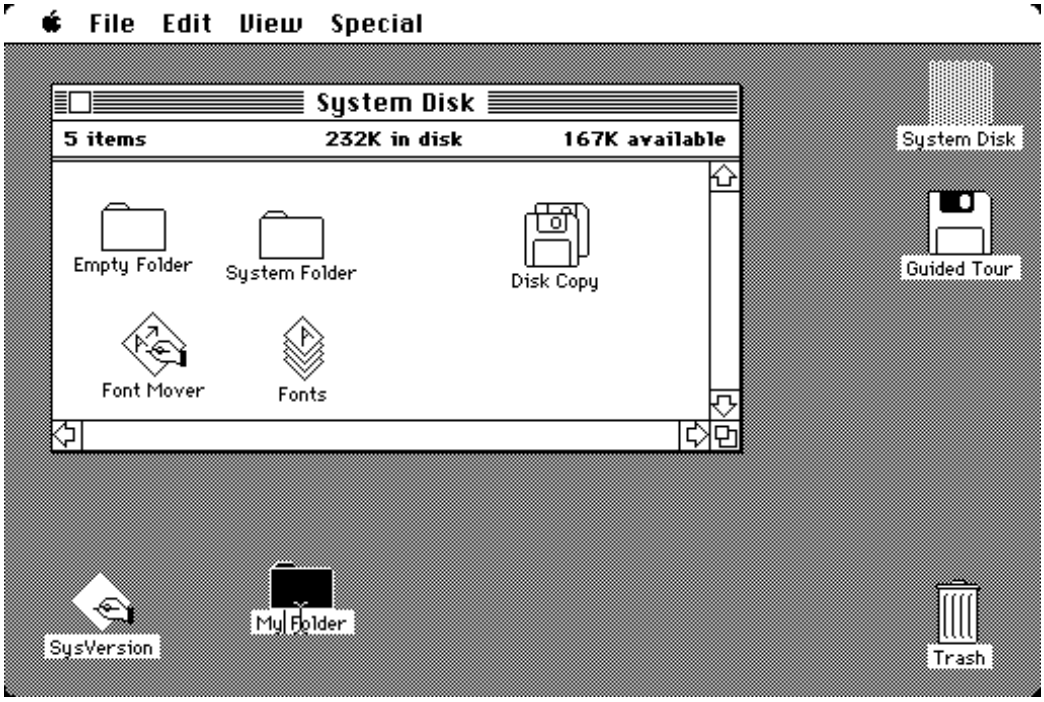
نتیجه؟ برای پیشگیری از مشکل ۵ب: به یاد داشته باشید که چرخ‌های آموزشی در دنیای لینوکس یک ابزار قابل نصب هستند و نه بخشی اصلی از سیستم. گاهی هم پیش می‌یاد که برنامه‌ای اصولا چیزی به اسم چرخ آموزشی ندارد.

مشکل ۶: تقلید در برابر همگرایی

یکی از چیزهایی که همیشه پیش می‌آید این است که وقتی آدم‌ها کشف می‌کنند لینوکس دقیقا همان ویندوز نیست، شروع می‌کنند به گفتن اینکه لینوکس اصولا از اول دقیقا قرار بوده همین باشد و اگر هم کسی سعی کند بخشی از آن را شبیه ویندوز کند، در اشتباه است. استدلال مخالف هم می‌شود: لینوکس یک صفحه متنی خط فرمان بوده و همین که الان گرافیکی است یعنی مشغول کپی کردن از ویندوز است تئوری بانمک، ولی اشتباهی است: اولین `X Windowing System` در سال ۱۹۸۴ نوشته شد که ادامه راه سیستم `W Windowing System` بود که پیش از این در دنیای یونیکس وجود داشت. ویندوز نسخه ۱، در سال ۱۹۸۵ منتشر شد که تازه تا سال ۱۹۹۰ که نسخه ۳ ویندوز آمد، کاربرد خاصی نداشت. در ۱۹۹۰ سیستم ایکس، به `X11` رسیده بود که تقریبا همان چیزی است که این روزها هم استفاده می‌شود. توجه کنید که پروژه لینوکس تازه در سال ۱۹۹۱ شروع شد. نتیجه؟ لینوکس `GUI` (رابط گرافیکی) را از ویندوز کپی نکرده: لینوکس از رابط گرافیکی‌ای استفاده کرده که مدت‌ها پیش از ویندوز، وجود داشته.

ویندوز ۳، جایش را به ویندوز ۹۵ داد که تغییری عظیم در رابط گرافیکی مایکروسافت بود. این نسخه تعداد زیادی ابداع و خصوصیت جید داشت: درگ و دراپ (بگیر/بنداز)، نوارابزار و ... . لینوکس هم کمی بعد،

این قابلیت‌ها را به خودش اضافه کرد...



اپل در ۱۹۸۴

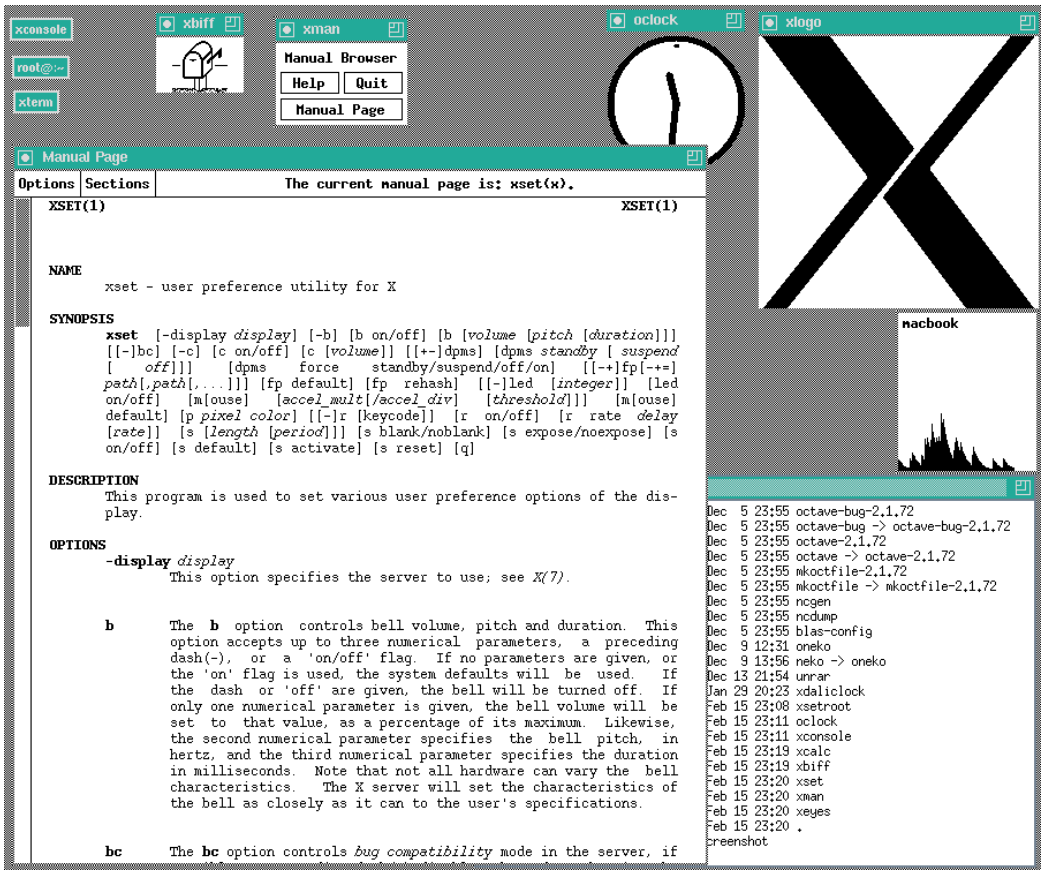
... نه (: همه این قابلیت‌ها و خصوصیت‌های جدید ، از قبل هم وجود داشته‌اند. NeXTSTEP یا قدم بعدی که عموی سیستم عامل امروزی اپل / مک محسوب می‌شود، تمام این قابلیت‌ها را داشت و نسخه اول آن‌ها مدت‌ها پیش از ویندوز ۹۵، یعنی در سال ۱۹۸۹ نوشته شد. این سیستم عامل آخرین نسخه خود را در سال ۱۹۹۵ به بازار عرضه کرد.

قبول قبول! مایکروسافت این قابلیت‌هایی که در ویندوز هست را خودش کشف نکرده اما به هرحال یک شکل و شمایلی به ویندوز داده که لینوکس سعی می‌کند آن را کپی کند برای بحث در این مورد، بهتر است سراغ بحث همگرایی تکاملی برویم. این بحث می‌گوید که دو موجود کاملا مستقل و مجزا، احتمالا در طول روند تکاملی به هم شبیه می‌شوند. در زیست شناسی این موضوع کاملا شناخته شده است. مثلا به کوسه و دلفین نگاه کنید. هر دو موجودات دریایی ماهیخوار هستند و اندازه ای مشابه دارند. هر دو باله پشتی دارند. هر دو باله‌های کناری دارند، هر دو روی دم‌هایشان باله دارند و فرم بدنی‌شان یک چیز است.

اما کوسه‌ها از ماهی تکامل یافته‌اند و دلفین‌ها از پستانداران چهارپای خشکی. دلیلی شباهت بیش از حد آن‌ها به همدیگر این است که هر دو گونه سعی کرده‌اند در حین تکامل به بالاترین بهره‌وری یک موجود دریازی

برسند. هیچ پیشا-دلفینی، به کوسه‌ها نگاه نکرده و نگفته «اوه... به این باله‌ها نگاه کن! چقدر خوبند. من هم باید یک باله برای خودم دست و پا کنم».

به همین روش به مدیرپنجره‌های اولیه لینوکس مثل FVWM و TWM نگاه کنید و ببینید چطور به نمونه‌های امروزی‌تری مثل Gnome و KDE با همه تسک‌بارها و آیکون‌ها و جینگولک‌بازی‌های امروزی مزین شده‌اند. هیچ شکی هم نیست که این تغییرات شبیه تغییراتی است که در ویندوز حاصل شده. خود ویندوز هم دقیقا همنطور است. ویندوز ۳ هم نه تسک بار داشت و نه دگمه استارت.



یونیکس مجهز به X در ۱۹۹۰

لینوکس در ابتدا میزکاری شبیه به میز کار ویندوزهای امروزی نداشت. مایکروسافت هم نداشت. حالا هر دو دارند. ایا این نشان دهنده چیزی به جز تکامل است؟ این به ما می‌گوید که هر دو اردوگاه برای رسیدن به بهره‌وری بالاتر رابط گرافیکی کار کرده‌اند و راه حل‌های مشابهی را انتخاب کرده‌اند.

مشکل ۷: جریان نرم‌افزار آزاد و بازمتن یا همان FOSS

اینجا یکسری مشکل هست. بحث این است که نرم‌افزار آزاد و بازمتن چیز فوق‌العاده‌ای است و در دنیای لینوکس یک مفهوم محوری است. درک دقیق این مفهوم و فهمیدن تفاوت آن با نرم‌افزارهای بسته و انحصاری، استدلالی کافی برای خیلی‌ها است که به دنیای لینوکس قدم بگذارند.

در بحث‌های قبلی به چند مورد از این تفاوت‌ها اشاره کرده‌ایم. مثلا اینکه در اینجا مردم به اشتباه فکر می‌کنند که دیگران وظیفه دارند به آن‌ها کمک کنند. اما بحث از این گسترده‌تر است.

شعار تبیین کننده هدف مایکروسافت این است «یک کامپیوتر روی هر میز کار» و مشخص است که این کامپیوتر باید از ویندوز استفاده کنند. هم مایکروسافت و هم اپل، سیستم عامل می‌فروشند و تمام تلاش آن‌ها این است که مردم بیشتر و بیشتر از محصولات آن‌ها استفاده کنند. این شرکت‌ها، تجاری هستند و به دنبال پول بیشتر. اما ما فاس (نرم افزار آزاد و بازمتن) را داریم که حتی امروز هم در بیشتر موارد غیرتجاری است.

قبل از اینکه ایمیل بزنید یا کامنت بگذارید و در مورد زوزه و ردهت و لینسپایر و .. بگویید، خودم باید بنویسم که که می‌دانم این شرکت‌ها لینوکس «می‌فروشند». آن‌ها دوست دارند که لینوکس در تمام دنیا مشهور شوند، آن هم لینوکس خاص خوشان. اما نباید ارائه دهنده‌ها را با سازنده‌ها قاطی کنیم. هسته (کرنل) لینوکس در یک شرکت نوشته نشده و کسی هم نمی‌خواهد از آن پول در بیاورد. ابزارهای گنو ، تجاری نیستند و توسط آدم‌هایی که سود مادی از اینکار نمی‌برند، توسعه داده می‌شوند. سیستم X11 که تا امروز، مشهورترین فراهم کننده امکانات گرافیکی است هم همینطور. در مورد نرم‌افزارهای کاربردی و میزهای کار هم همینطور: گنوم،فلاکس باکس، اینلایت‌منت و غیره. می‌بینم که آدم‌هایی هستند که در دنیای لینوکس کار تجاری بکنند ولی فعلا در اقلیتند.

زیاد شدن استفاده از نرم‌افزارهای انحصاری و تجاری به معنی سود مستقیم توسعه دهندگان آن در شرکت‌های صاحب نرم‌افزار است. در فاس، وضع اینطور نیست. اینجا هیچ توسعه دهنده آزاد و بازمتنی از استفاده شدن برنامه‌اش سود مستقیم نمی‌برد. اما سودهای جنبی همیشه وجود دارد: افتخار شخصی، احتمال کشف بیشتر باگ‌ها، احتمال جذب توسعه‌دهندگان جدید، احتمال دریافت پیشنهاد شغل بهتر و ...

لینوس تورووالدز از اینکه مردم بیشتر و بیشتر از لینوکس استفاده کنند، استفاده مادی نمی‌برد. ریچارد استالمن با بیشتر شدن استفاده گنو، پولدار نمی‌شود. تمام سرورهایی که از اپن بی اس دی و اپن اس اس اچ نصب می‌کنند، یک ریال هم نصیب پروژه اوپن بی اس دی نمی‌کنند. این ایجاد کننده یکی از مشکلات پیچیده دنیا است:

تازه واردهای دنیای بازمتن کشف می‌کنند که کسی مشتاق دیدن آن‌ها نیست.

تازه واردها، از دنیاهایی می‌آیند که در آن‌ها اصلی‌ترین هدف سیستم عامل‌ها، «کاربرپسند» بودن و «تمرکز بر مشتری» بوده و حالا ناگهان کشف می‌کنند سیستم عاملی که تازه به سراغ آن آمده‌اند هنوز از صفحات man به عنوان راهنما استفاده می‌کند و تنظیمات اصلی آن از طریق چند فایل متنی و جستجو در گوگل قابل

تغییرند. اگر این آدم‌ها شروع به غر زدن کنند، کسی از آن‌ها عذر خواهی نمی‌کند و حتی ممکن است خیلی سرراست، آن‌ها را به در خروجی راهنمایی کنند.

بله! کمی اغراق کردم. ولی مطمئن هستم اگر با کسانی که به لینوکس آمده و برگشته‌اند گپ بزنید، این چیزی است که از خیلی از آن‌ها خواهید شنید.

به یک مفهوم عجیب، آزادی و بازمتنی روشی بسیار خودخواهانه در توسعه نرم‌افزار است: مردم فقط روی چیزهایی که دوست دارند کار می‌کنند. از نظر اکثریت این آدم‌ها، نیازی نیست لینوکس برای تازه واردها جذاب شود: همین الان لینوکس اکثر چیزهایی که خود این آدم ها می‌خواهند را به خوبی انجام می‌دهد و دیگر چه نیازی است که برای نیازهای دیگران تلاش کنیم؟

فاس در بسیاری از جنبه‌ها شبیه اینترنت است: شما به کسی که وبلاگ/وبسایت/نرم‌افزار می‌نویسد پول نمی‌دهید تا آن را بخوانید/نصب کنید. یک روند ساده برای وصل شدن به اینترنت/رابط گرافیکی برای کسی که در حال حاضر به اینترنت متصل است/نرم‌افزار را استفاده می‌کند ارزش چندانی ندارد. نویسندگان/برنامه‌نویسان نیازی ندارند خواننده/کاربر زیادی داشته باشند تا وبلاگ/برنامه شان را بنویسند. خیلی‌ها هستند که از این کارها پول در می‌آورد ولی نه به شیوه قدیمی «این مال من است و برای استفاده باید پول بدهی» بلکه این آدم‌ها از تبلیغ و تجارت الکترونیکی / پشتیبانی پول درمی‌آورند.

لینوکس علاقه‌ای به سهم بازار ندارد. لینوکس چیزی به اسم مشتری ندارد. لینوکس برای پول درآوردن نوشته نشده. هدف لینوکس این نیست که پرکاربردترین سیستم عامل سیاره شود.

چیزی که لینوکسی‌ها می‌خواهند یک سیستم عامل خوب، آزاد و پرقابلیت است. اگر این مساله باعث شود که لینوکس پرکاربردترین سیستم عامل جهان شود، عالی است. اگر این باعث شود که لینوکس زیباترین و راحت‌ترین رابط کاربری را داشته باشد، عالی است. اگر هم این باعث شود که لینوکس یک بازار تجاری چند میلیارد دلاری درست کند، عالی است.

اینها **عالی** هستند ولی **هدف** نیستند. هدف این است که لینوکس بهترین سیستم‌عاملی شود که جامعه‌اش توان نوشتن‌اش را دارند. آنهم نه برای دیگران، برای خوشان. عبارت‌های مرسومی مثل اینکه «لینوکس هیچگاه سیستم‌عامل اول جهان نخواهد شد مگر اینکه فلان کار را بکند» اصولا عبارتی بی‌ربط است. متنفرین از مایکروسافت و لینوکس پرستان و شرکت‌های تجاری که از لینوکس پول در می‌آورند، ممکن است پر سر و صدا باشند ولی در دنیای لینوکس، اقلیتند.

این چیزی است که جامعه لینوکس می‌خواهد: سیستم‌عاملی که هر کس که آن را بخواهد، بتواند نصبش کند. پس اگر به مهاجرت به لینوکس فکر می‌کنید، دقیقا به این فکر کنید که **شما** دنبال چه چیزی هستید.

اگر دنبال سیستم‌عاملی هستید که دست شما را نبندد، دستتان را باز بگذارد و اجازه بدهد در صندلی راننده بنشینید و انتظار داشته باشد که شما می‌فهمید دارید چکار می‌کنید ، لینوکس انتخاب خوبی است. برای استفاده از این سیستم عامل باید کمی وقت صرف کنید اما وقتی کمی آن را یاد گرفتید، هر کار که دوست

دارید از عهده‌اش بر خواهد آمد.

اگر فقط به دنبال یک ویندوز هستید که مشکل امنیتی نداشته باشد و ویروس نگیرد سراغ چند راهنمای کامپیوتر بروید و برنامه ضدویروس درستی نصب کنید که فایروال بگذارید و مواظب بدافزارهای باشید و به جای اینترنت اکسپلورر از یک برنامه امن‌تر مثل فایرفاکس استفاده کنید و همه به روزرسانی‌های امنیتی را نصب کنید. آدم‌هایی هستند (از جمله خودم) که از زمان ویندوز ۳.۱ تا به XP از آن استفاده کرده‌اند بدون اینکه یکبار مشکل حاد ویروس یا دردسر امنیتی داشته باشند. شما هم می‌توانید همین احساس را تجربه کنید. اگر به دنبال یک ویندوز بدون ویروس هستید، لینوکس شما را ناامید خواهد کرد.

اگر هم دنبال ایمنی و قدرت یونیکس هستید و در عین حال تمرکز بر مشتری و کاربرپسندی و پشتیبانی تجاری را هم می‌خواهید، به اپل و مک فکر کنید.

فصل ۲

لینوکس روزمره

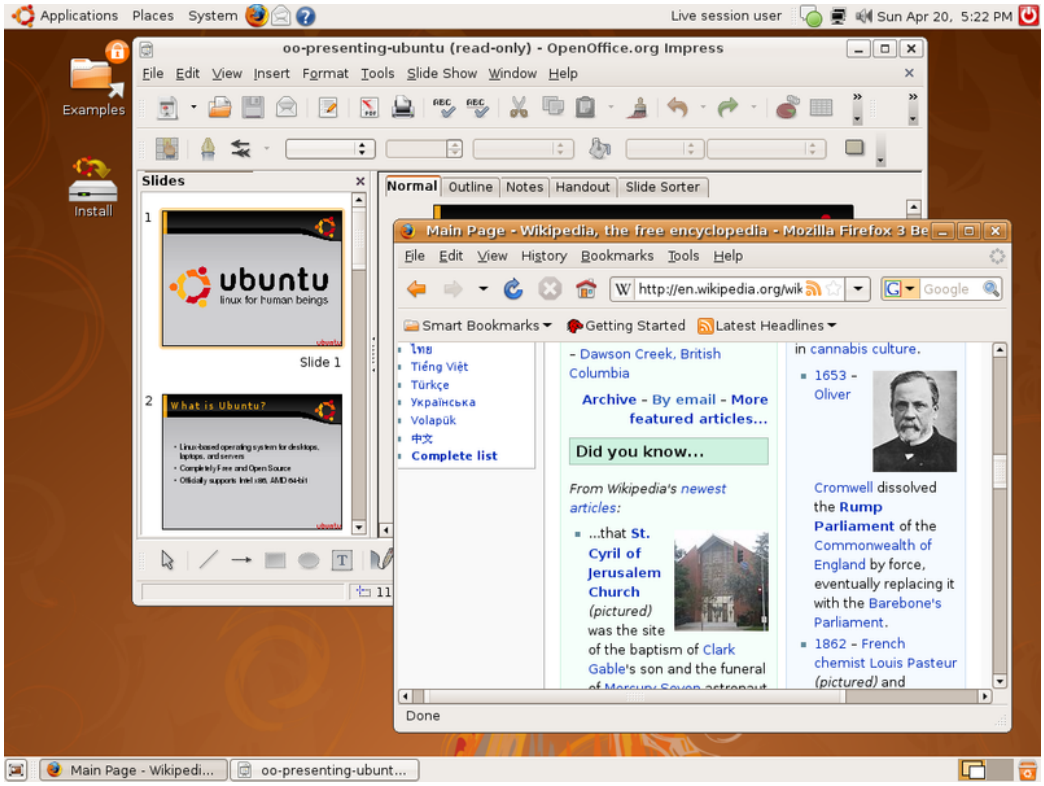
۱.۲ مفهوم توزیع و منابع

توزیع یا Distribution یکی از اولین چیزهایی است که بعد از تصمیم به امتحان لینوکس، خواهید شنید. بعضی‌ها از اوبونتو حرف می‌زنند، بعضی‌ها از اوپن زوزه و بعضی‌ها هم به شما پارسیکس را پیشنهاد می‌کنند. بدون شک حینی که کسی دارد به شما اوبونتو را پیشنهاد می‌کند، یک نفر وسط حرفش خواهد پرید و پیشنهاد خواهد داد از پی سی او اکس لینوکس یا لینوکس مینت که ساده‌تر است استفاده کنید.

معمولا این بحث‌ها خیلی گیج کننده می‌شوند و در نهایت هم فقط باعث گمراهی و گیج شدن. اما واقعا بحث سر چیست؟

توزیع

از بخش‌های قبلی، به یاد دارید که لینوکس فقط یک هسته است و سیستم‌عاملی که ما واقعا استفاده می‌کنیم باید گنو/لینوکس نامیده شود. در واقع هسته‌ای به اسم لینوکس وجود دارد که بعد از ترکیب با مجموعه عظیمی از نرم‌افزارهای دیگر، یک سیستم عامل مدرن مثل این را تشکیل می‌دهد.



در واقع شما اگر شخصا بخواهید از صفر یک سیستم لینوکس راه بیاندازید، باید اول شخصا هسته را

کامپایل کنید و بعد از طریق نصب یک مدیر بوت (مثلا گراب) به کامپیوتر بگویید که بعد از نصب آن هسته را لود کند و علاوه بر این، تمام برنامه‌های مورد نظرتان (مثلا خط فرمان، محیط گرافیکی، ماشین حساب، آفیس و ...) را هم کامپایل و به همدیگر متصل کنید.

این کار سختی است و حتی برای کسی که کاملا آن را بلد باشد، حداقل چندین روز به طول می‌کشد. بله (: چندین روز. اما در اکثر مواقع نیازی به این همه دردسر نیست. شرکت‌ها و افراد علاقمند یکبار تمام این کارها را کرده‌اند و با اضافه کردن یک روند نصب معمولا گرافیکی و راحت، یک «توزیع» لینوکس برای شما ساخته‌اند.

مثلا اگر شما به سراغ توزیع اوبونتو بروید، دیسکی را در دست خواهید داشت که مهندسان شرکت کانونیکال سر هم کرده‌اند. آن‌ها یک هسته مناسب و تست شده را به علاوه یک میزکار و کلی برنامه مفید کامپایل و تنظیم کرده‌اند که شما با پیگیری یک روند نصب ساده و گرافیکی می‌توانید آن را روی کامپیوتر خود نصب کنید. اگر به جای اوبونتو، توزیع جذابی مثل پارسیکس را انتخاب کنید، با یک سی دی روبرو خواهید بود که آلن باغومیان از کنار هم چیدن هسته و برنامه‌هایی که خودش ترجیح داده ساخته و کاری کرده که زبان فارسی به شکل پیش‌فرض در سیستم فعال باشد.

می‌بینید که توزیع چیز خیلی عجیبی نیست. واقعا هم توزیع‌ها چیزهای عجیبی نیستند بلکه مجموعه‌ای هستند از چند برنامه و یک هسته و کمی تنظیمات. به عبارت دیگر، شما می‌توانید هر برنامه‌ای را در هر توزیعی نصب کنید و انتخاب یک توزیع فقط نشان دهنده برنامه‌ها و تنظیمات پیش فرض است.

البته این را هم به یاد داشته باشید که به جز بسته‌ها و تنظیمات پیش‌فرض، چند چیز دیگر هم با انتخاب توزیع، تغییر می‌کنند از جمله مدیریت بسته و فلسفه توزیع.

مدیریت بسته

همانطور که بالاتر خواندید، یک سیستم عامل یونیکسی (مثل گنو/لینوکس) مجموعه‌ای است از بسته‌ها. حالا فرض کنید بخواهید یک بسته به سیستم خود اضافه کنید یا از آن کم کنید. برای اینکار معمولا یک یا چند برنامه ویژه وجود دارند که وابسته به توزیعی هستند که انتخاب می‌کنید. مثلا همه توزیع‌های مبتنی بر دبیان (مثل اوبونتو و پارسیکس)، از مدیر بسته‌ای به نام `apt` و در سطح فنی‌تر `dpkg` استفاده می‌کنند در حالی که توزیع‌های مشابه ردهت مدیر بسته‌های مبتنی بر `rpm` دارند. در این مورد در بخش انتخاب توزیع بیشتر صحبت خواهیم کرد.

فلسفه توزیع

خیلی‌ها دوست دارند از این شعار استفاده کنند که «گنو/لینوکس فقط یک سیستم‌عامل نیست» (: واقعا هم اینطور است. نرم‌افزار آزاد یک فلسفه قوی با پشتوانه نظری و اجتماعی است و لینوکس‌های مختلف هم

دیدگاه‌های مختلفی نسبت به جهان دارند. بعضی‌ها مانند آرچ لینوکس معتقد به اصل سادگی هستند، بعضی‌ها مثل دبیان معتقد به پایداری و بعضی‌ها مثل مینت معتقد به کاربر پسند بودن. شما با انتخاب یک توزیع، فلسفه آن توزیع را هم انتخاب می‌کنید و این فلسفه‌ها با هم تفاوت می‌کنند. برای بعضی‌ها در حد فلسفی و اخلاقی درست است که از یک توزیع کاملا پایبند به اصول آزادی نرم‌افزار مثل گنوسنس استفاده کنند و بعضی‌ها هم به شما اصرار خواهند کرد که از لینوکس‌هایی مثل مینت استفاده کنید که با صرف نظر کردن از بخش‌هایی از فلسفه آزادی نرم‌افزار، کاربر پسندتر شده و مثلا فایل‌های صوتی تصویری انحصاری را بدون هیچ شکایتی پخش می‌کند.

۲.۲ معماری های لینوکس

معمولا وقتی میزکار و توزیع خود را انتخاب کردید، می‌توانید به سراغ دانلود دیسک و نصب لینوکس بروید اما گاهی یک سوال کمی حرفه‌ای تر شما را گیج می‌کند: ۳۲ بیت یا ۶۴ بیت.

از قدیم تقریبا تمام توزیع‌های مشهور لینوکس هم با نسخه ۳۲ بیت عرضه می‌شدند و هم به شکل نسخه ۶۴ بیتی و اینکه کاربر کدام را دانلود و نصب می‌کرد وابسته به فاکتورهای مختلفی بود. تا یکسال پیش تقریبا اکثر توزیع‌های بزرگ به شکل پیش‌فرض شما را به سمت نسخه‌های ۳۲ بیتی راهنمایی می‌کردند و حالا توزیع‌هایی مثل اوبونتو و فدورا در سایت‌هایشان نسخه‌های ۶۴ بیتی را گزینه پیش فرض قرار داده‌اند. اما اول بگذارید ببینیم مفهوم این عددها چیست.

جریان چیست؟

عددهایی مثل ۳۲ یا ۶۴ نشان دهنده اندازه (عرض) حافظه‌ای است که یک پردازشگر (سی پی یو) می‌تواند به آن دسترسی پیدا کند. اگر بخواهیم عبارت را دقیق‌تر بیان کنیم باید بگویم که گفتن عبارت «کامپیوتر ۶۴ بیتی» یعنی پردازشگر این کامپیوتر رجیسترهایی به اندازه ۶۴ بیت داشته و در هر عملیات واحد می‌تواند روی ۶۴ بیت پردازش انجام دهد.

مشخص است که یک کامپیوتر ۶۴ بیتی با داشتن رجیسترهای بزرگتر به راحتی می‌تواند تمام دستورات و برنامه‌های ۳۲ بیتی را اجرا کند ولی معکوس این جریان صادق نیست. پس اگر شما یک کامپیوتر ۶۴ بیتی دارید مجاز هستید روی آن سیستم عامل ۳۲ بیتی یا ۶۴ بیتی نصب کنید ولی اگر از یک کامپیوتر ۳۲ بیتی استفاده می‌کنید، نباید به سراغ نسخه ۶۴ بیتی سیستم عامل‌ها بروید. البته واقعا نیاز به استرس نیست، خبر خوش من این است : به احتمال خیلی زیاد شما یک کامپیوتر ۶۴ بیتی دارید.

فهرست پروسسورهای ۶۴ بیت

چون در دنیای ویندوز سیستم‌عامل‌های ۶۴ تازه برای کارهای دسکتاپ مشهور شده‌اند، عده‌ای هیجان دارند که درباره آن صحبت کنند اما واقعیت این است که مفهوم پروسسور ۶۴ بیتی مفهومی کاملا جا افتاده و حتی قدیمی است و اگر کامپیوتر شما از سال ۲۰۰۸ به بعد ساخته شده، به احتمال زیاد معماری ۶۴ بیتی دارد. فهرست زیر نشان دهنده پردازنده‌های ۶۴ بیت موجود است:

```
AMD:
Athlon64, Athlon FX, Athlon X2, Phenom, Semprons that
use AM2/AM2+/AM3 socket, Turion64
Intel:
F and 5x1 series Pentium 4 using the ”Prescott” core,
Pentium D, Core 2 (Solo, Duo & Quad), Core i3 (all),
Core i5 (all), Core i7 (all), VIA, Isiah
```

و احتمالا هر چیزی که در آینده ساخته شود هم از همین معماری بهره خواهد برد. در صورتی که همین حالا هم پشت یک کامپیوتر لینوکسی نشسته‌اید، با زدن دستور زیر می‌توانید پهنای رجیسترهای خود را بررسی کنید:

```
grep –color=always -iw lm /proc/cpuinfo
```

در صورتی که خروجی این دستور حاوی `lm` های رنگی باشد، می‌گوید که سی‌پی‌یو(ها)ی شما از حالت `long mode` یا همان آدرس دهی ۶۴ بیتی پشتیبانی می‌کنند.

مقایسه سرعت

بر اساس تست‌های انجام شده و عقل سلیم، یک سیستم ۶۴ بیت سریعتر از یک سیستم ۳۲ بیت کار می‌کند. اما احتمالا برایتان جالب است که بگویم تقریبا هیچ انسان عادی نمی‌تواند تفاوت سرعت اجرای برنامه‌های معمول در دو حالت ۳۲ و ۶۴ بیت را تشخیص بدهد چون تست‌ها به عددهای بین ۵ تا ۱۵ درصد سریعتر شدن برنامه‌ها اشاره می‌کنند. مساله این است که برنامه‌های ۶۴ بیت دقیقا همان برنامه‌های ۳۲ بیت هستند که دوباره کامپایل شده‌اند و هیچ بهینه‌سازی خاصی برای استفاده از تمام قابلیت‌های پروسسور در آن‌ها صورت نگرفته. پس اگر فکر می‌کنید با نصب ۶۴ بیت به سرعت خیلی بهتری می‌رسید، تجدید نظر کنید.

حافظه

از نظر ریاضی یک سیستم ۳۲ بیتی می‌تواند تا ۴ گیگابایت حافظه و یک سیستم ۶۴ بیت تا ۸.۱۶ میلیون ترابایت (۱۶ اگزابایت) حافظه را آدرس دهی کند. پس اگر شما ۸ گیگابایت رم داشته باشید، در حالت عادی یک سیستم ۳۲ بیتی فقط خواهد توانست از ۴ گیگ آن استفاده کند در حالی که یک سیستم ۶۴ می‌تواند میلیون‌ها برابر بزرگتر از آن را هم کنترل کند. اما این اصلا به این معنی نیست که اگر بیشتر از ۴ گیگ رم دارید، ملزم به نصب ۶۴ بیت هستید. از مدت‌ها پیش، یک افزونه در کرنل به نام `Physical Address`

Extention که به شکل مخفف PAE خوانده می‌شود، محدودیت آدرس دهی کرنل‌های ۳۲ بیتی را به ۶۴ گیگ افزایش داده.

اگر بیشتر از ۴ گیگابایت رم دارید و می‌خواهید از یک سیستم عامل (و در نتیجه کرنل ۳۲ بیتی) استفاده کنید باید مطمئن شوید که کرنل pae را نصب کرده‌اید. معمولا می‌توانید این کرنل را در مخازن با جستجو به دنبال pae پیدا کرده و نصب کنید. با توجه به طبیعی‌تر شدن داشتن رم‌های بالای ۴ گیگ، بعضی توزیع‌های مشهور (از جمله اوبونتو) در نسخه‌های اخیر خود به شکل پیش فرض از کرنل PAE استفاده می‌کنند.

سازگاری

تقریبا تا دو سال قبل، استفاده از یک لینوکس ۶۴ بیت با دردسرهایی مثل سخت تر بودن نصب بعضی برنامه‌ها (بخصوص فلش و جاوا) همراه بود. این مشکل حالا تا حد زیادی مرتفع شده. البته کماکان نصب یک سیستم ۶۴ بیتی ممکن است در لحظاتی باعث دردسرهایی در نصب برنامه های جدید بشود. مثلا برنامه اسکایپ یا یک بازی جدید که توسط یک کمپانی به شکل باینری برای لینوکس عرضه شده ممکن است روی سیستم‌های ۳۲ بیت فقط با دانلود و دبل کلیک کردن اجرا شود اما برای نصب روی یک سیستم ۶۴ بیت نیازمند درک / کار بیشتری باشد (نصب کتابخانه‌های ۳۲ بیتی سازگار کننده). اما این واقعا به ترسناکی قدیم نیست و حداقل انتظار می‌رود که تمام برنامه‌های معمول روی هر دو نسخه عرضه شوند.

باحال بودن. ۶۴ بیت برای خیلی‌ها یک مفهوم جدید است و هیجان انگیز. آدم‌ها دوست دارند کارهای جدید بکنند و «متفاوت» باشند و در نتیجه نصب و استفاده از یک سیستم ۶۴ بیت با اینکه در عمل هیچ تفاوتی با نصب و استفاده از یک سیستم ۳۲ بیت ندارد، برای عده‌ای کشش خاص خودش را دارد.

و حرف نهایی

برای یک کاربر خانگی رومیزی، تفاوت بزرگی بین سیستم‌های ۳۲ بیت و ۶۴ بیت وجود ندارد. اگر سیستم ۳۲ بیت نصب می‌کنید و بیشتر از چهار گیگابایت رم دارید باید مطمئن شوید که از کرنل pae استفاده می‌کنید تا تمام حافظه‌تان استفاده شود و اگر سیستم ۶۴ بیت نصب می‌کنید باید آماده باشید که یک روز یک برنامه بسته برایتان کمی دردسر درست کند و مجبور بشوید سراغ نصب چند لایبری ۳۲ بیت از مخازنتان بروید.

اگر هم نظر من را می‌خواهید باید بگویم که دوباره نوشته‌های بالا را بخوانید و خودتان انتخاب کنید چون این دو گزینه تقریبا مساوی همدیگر هستند. یک راه حل هم این است که اگر هیچ ایده‌ای ندارید از سیستمی استفاده کنید که سایت دانلود رسمی همان توزیع به شما پیشنهاد می‌دهد.

نکات:

- صفحه ویکی جامعه اوبونتو در مورد انتخاب بین ۳۲ و ۶۴ بیت را ببینید[۱].
- در دنیای لینوکس کرنل‌های ۶۴ بیتی معمولا با پسوند 64_ و سیستم‌عامل‌های ۳۲ بیتی با پسوندهایی مانند i686 یا i386 شناخته می‌شوند
- بر اساس خروجی این دستور می‌بینید که من (جادی) از یک کرنل ۳۲ بیتی (i686) با قابلیت دسترسی به حافظه بیشتر از ۴ گیگابایت (PAE) و علیرضا (cyletech) از یک کرنل ۶۴ بیتی استفاده می‌کنیم.

```
$ uname -a
Linux jedora 3.6.9-2.fc17.i686.PAE 1 SMP Tue Dec 4
14:15:28 UTC 2012 i686 i686 i386 GNU/Linux
Linux cyletech 3.5.0-34-generic 55-Ubuntu SMP Thu Jun 6
20:18:19 UTC 2013 x86_64 x86_64 x86_64 GNU/Linux
```

[۱] help.ubuntu.com

۳.۲ انتخاب دسکتاپ و توزیع

ادغام با بخش مفهوم توزیع

انتخاب توزیع یکی از اولین مشکلاتی است که تازه واردان به لینوکس با آن رو به رو می‌شوند. انتخاب یک توزیع از بین تقریبا ۲۰۰۰ توزیعی که در سایت دیستروواچ از آن‌ها نام برده شده، کار خیلی خیلی سختی است، حتی اگر توزیع‌ها را به تعداد خیلی کمتری محدود کنیم، بازهم انتخاب خیلی سخت خواهد بود.

مساله وقتی پیچیده‌تر می‌شود که از یک گروه لینوکس کار بپرسید «کدام توزیع را انتخاب کنم؟». این سوال معمولا منجر به چیزی می‌شود که جامعه لینوکس به آن جنگ توزیع‌ها می‌گویند: تعداد بسیار زیادی نظر که پشت سر هم می‌آید و هر کدام ادعا دارند که فلان توزیع بهترین توزیع است و منجر به ایمیل‌های تندتر بعدی می‌شود که سعی می‌کنند بگویند چرا فلان توزیع بهترین نیست و در نهایت نظرات ناشی از عصبانیتی که حتی در مواردی خواندنشان آدم را از آمدن سراغ لینوکس پشیمان می‌کند.

خب ... پس جواب چیست؟ اگر گفتن اینکه «فلان توزیع بهترین توزیع است» باعث دردسر می‌شود پس جواب «بهترین توزیع» چیست؟ مساله این است که جواب خیلی ساده است، این سوال است که سخت است. مثلا در مورد من و لپ تاپم در همین لحظه سوال ممکن است چیزی شبیه به این باشد:

> من تا دیروز یک لینوکس داشتم که از آن راضی نبودم چون راحت نمی توانمستم برنامه‌های مورد نظرم را در آن نصب کنم و در نتیجه چند روزی است که از کارها عقب مانده‌ام و در این لحظه باید چیزی نصب کنم که بتوانم در کمترین زمان ممکن سیستم را به حالت قابل استفاده برسانم. من قبلا مدت‌های خیلی طولانی از اوبونتو دقیقا در این محیط استفاده کرده ام و در حال حاضر هم آخرین نسخه اوبونتو، نسخه ۹.۱۰ است.

دیدید؟ این سوال است که سخت است ولی جواب کاملا ساده و مشخص است: اوبونتو.

مساله توزیع، کاملا وابسته به افراد و شرایط خاص آن‌ها است، اگر کسی به جز این گفت، مطمئن باشید که نسبت به توزیعش تعصب دارد. در عین حال، مساله مهمتر این است که شما باید لینوکس بلد باشید، نه توزیع. توزیع‌های لینوکس خیلی به هم نزدیک هستند، در واقع همانطور که در فصل قبل گفتیم، یک توزیع چیزی بیشتر از یکسری برنامه نیست و تقریبا هر توزیع مهمی را می‌شود با کمی تنظیمات به راحتی شبیه به محیط توزیع دیگر کرد.

برای انتخاب یک توزیع، اول از همه باید هدف خود را بشناسید. اگر دنبال توزیعی برای یاد گرفتن لینوکس هستید، از فهرست زیر فراتر نروید:

- پارسیکس
- اوبونتو

- اوپن زوزه
- فدورا
- دبیان
- مینت
- مندرویوا
- ...

(: هاها.. دیدید؟ هیچ وقت حاضر نشوید کسی شما در یک لیست محدود کند. هر چیزی که دوست دارید را امتحان کنید ولی بدانید که وقتی سراغ توزیع‌های کمتر مشهور می‌روید (مثلا سورس میج)، کمک کمتری خواهید داشت و احتمال اینکه توزیع آنطور که قرار است کار نکند هم بیشتر است. برای شروع همیشه خوب است که یکی از توزیع‌های در دسترس را امتحان کنید. یک سی دی اوبونتوی زنده در درایو بگذارید و بعد از بوت شدن کامپیوتر، کمی با آن کار کنید. همین کار را با فدورا و اوپن زوزه و دبیان و مینت هم بکنید. در نهایت آنی که دوست دارید را نصب کنید و با آن پیش بروید.

این روزها توزیع‌های خانواده اوبونتو (از جمله خود اوبونتو، کوبونتو، مینت، ...) بسیار مشهور هستند و کاربران زیادی دارند. در عین حال این توزیع در ایران هم استفاده کننده زیاد و در نتیجه یک فروم فعال دارد و به این دلایل شروع با این توزیع کار شاید راحت‌تر از بقیه باشد. اما این اصلا به معنی بهتر بودن این توزیع نیست و سراغ هر توزیع دیگری هم که بروید بخش عمده‌ای از سوالات شما در همان فروم یا در فروم تخصصی لینوکس ایران جواب خواهد گرفت.

نکته نهایی هم این است که مواظب باشید تبدیل به یکی از همان‌هایی که در ابتدای مقاله ذکرشان رفت نشوید. فراموش نکنید که ما طرفدار گنو/لینوکس هستیم و آزادی نرم‌افزار و نه یک آدم متعصب نسبت به یک توزیع خاص. بعد از اینکه سراغ یک توزیع رفتید و به آن عادت کردید، حتما فرصتی به خودتان بدهید تا بقیه چیزها را هم انتخاب کنید. بخصوص توزیع‌های خانواده‌های دیگر را. اصلا جذاب نیست ادعای بلد بودن لینوکس بکنید و از مدیر بسته `rpm` هیچ چیز ندانید یا قهرمان مدیریت سیستم در ردهت باشید اما نتوانید یک سرویس را در دبیان به یک اینیت اضافه یا از آن حذف کنید.

به عنوان یک تازه کار، توزیع‌ها را امتحان کنید و بعد از انتخاب یکی از مشهورها برای شروع، آن را یاد بگیرید. اما فراموش نکنید که همیشه سراغ توزیع‌های دیگر هم بروید و با تفاوت‌ها آشنا شوید. روش رشد گنو/لینوکس دقیقا همان روش طبیعی است: انتخاب اصلح. مطمئن باشید که اگر توزیعی خیلی بد باشد، چندان طولانی زنده نخواهد ماند و اگر توزیعی واقعا بهتر از بقیه باشد، بقیه به سرعت جنبه‌های مثبت آن را به توزیع خودشان اضافه خواهند کرد.

۴.۲ استفاده از لینوکس با دیسک زنده

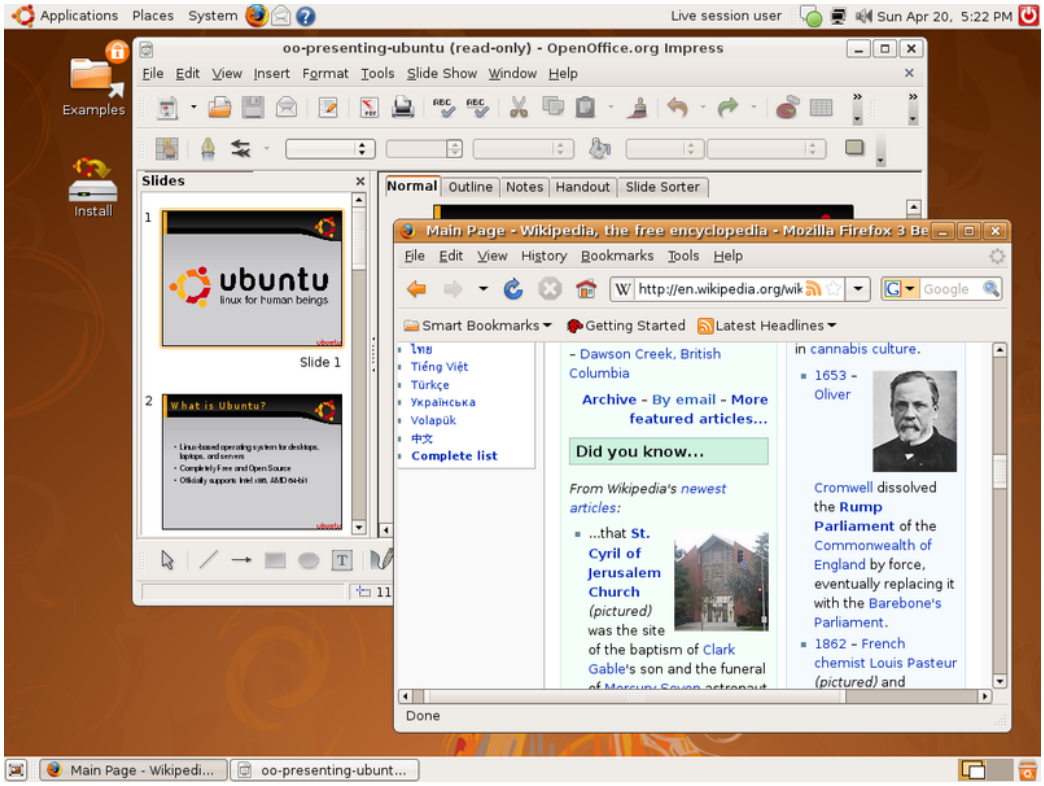
در زمان قدیم (و هنوز هم در مورد سیستم‌عامل‌هایی مثل ویندوز و مک) اگر کسی بخواهد از سیستم عامل استفاده کند، می‌بایست آن را نصب کند. اما چرا؟ مگر «بوت شدن» چیزی بیشتر از لود شدن کرنل و بعد اجرا شدن چند برنامه (مثلا محیط گرافیکی) است؟ این دقیقا سوالی بود که ایگدراسیل لینوکس هم در سال ۱۹۹۲ از خودش پرسید. ایگدراسیل در افسانه‌های نورث، درختی عظیم ست که نه جهان در سر شاخه‌های آن قرار گرفته‌اند. لینوکس ایگدراسیل هم سعی می‌کرد سیستم عاملی باشد که بشود آن را فقط از روی سی دی بوت کرد و سپس از سرشاخه‌های آن که برنامه‌های آزاد دیگر بودند، استفاده کرد.

اما پروژه ایگدراسیل شکست خورد و عرضه آن در سال ۹۵، متوقف شد. یکی از اصلی‌ترین دلایل شکست، سرعت کم درایوهای سی دی آن زمان بود. مفهوم دیسک زنده لینوکس از زمان خود بسیار جلوتر بود و به همین دلیل مجبور شد تا سال ۱۹۹۸ و افزایش سرعت سی‌دی درایوها و حافظه کامپیوترها منتظر بماند.

در ۱۹۹۸ بود که دمو لینوکس به عنوان اولین نسخه عملی یک لینوکس پا به عرصه وجود گذاشت. این نسخه می‌توانست تنها با گذاشتن یک سی دی در درایو و بدون هیچگونه تنظیمی، یک پی سی را در محیط گرافیکی بوت کند و با استفاده از سیستم فایل فشرده‌اش، امکان اجرا کردن صدها برنامه (از جمله استارآفیس) را بدون هیچ تغییری در هارد دیسک فراهم کند. این توزیع تا سال ۲۰۰۲ فعال بود و در آخرین نسخه‌ها امکان نصب بر روی هارددیسک هم به آن اضافه شده بود و این دقیقا چیزی است که راه را به اکثر توزیع‌های حال حاضر جهان نشان داده.

اما قبل از تمام کردن بحث تاریخی، باید ذکری هم از ناپیکس بکنیم. مشهورترین نسخه لینوکس زنده و توزیعی که تقریبا همه آن را به عنوان اولین دیسک زنده‌ای که در تمام جهان استفاده شد، می‌شناسند. ناپیکس توزیعی مبتنی بر دبیان بود که در ۲۰۰۳ رسما عرضه شد. این توزیع می‌توانست به طور گرافیکی یک سیستم را بوت کند و سپس به عنوان یک سیستم عامل برای نجات سیستم‌های مشکل پیدا کرده یا به عنوان یک سیستم عامل مستقل استفاده شود. بوت زنده ناپیکس، آینده تمام توزیع‌های لینوکس بود.

در حال حاضر تقریبا تمام توزیع‌های مشهور لینوکس، نسخه دیسک زنده (`Live disk`) هم دارند و کار تا جایی پیش رفته که می‌شود گفت اکثر توزیع‌های دسکتاپ، اصولا نسخه غیرزنده ندارند. مثلا در مورد اوبونتو، شما همیشه یک فایل `iso` دانلود می‌کنید که بعد از نوشتن آن روی دیسک (به شیوه گفته شده در تهیه یک نسخه از لینوکس) به یک دیسک زنده تبدیل می‌شود. کافی است این سی دی را در درایو یک پی سی که برای بوت از روی سی دی تنظیم شده (در بایوس) بگذارید و کامپیوتر را روشن کنید. مطمئنا به خاطر سرعت کم سی‌دی نسبت به هارد دیسک، بوت شدن کندتر از حالت طبیعی خواهد بود ولی بعد از یکی دو دقیقه، بدون هیچ تغییری در هارد دیسک کامپیوتر سیستم با لینوکس مورد نظر بوت خواهد شد. مثلا اگر از مندریوا استفاده کرده باشید، صفحه چیزی شبیه به این خواهد بود:



در این وضعیت شما می‌توانید برنامه‌های موجود را نصب و استفاده کنید، به سخت افزارها دسترسی داشته باشید، در اینترنت گشت بزنید، چت کنید و خیلی کارهای دیگر بدون اینکه به سیستم عامل اصلی نصب شده روی هارد دیسک صدمه‌ای بخورد. دیسک زنده کاربردهای متنوعی دارد. از جمله:

- انجام کارهایی که در سیستم عامل خاصی می‌شود انجام داد: مثلا بازیابی اطلاعات هارددیسک یا پاک کردن ویروس‌های یک درایو ویندوزی با استفاده از دیسک زنده لینوکس.

- آماده شدن برای نصب: بسیاری از توزیع‌های جدید برای نصب شدن از نسخه زنده استفاده می‌کنند.

- بررسی سازگاری سخت افزاری: شما می‌توانید قبل از نصب هر نسخه‌ای از لینوکس، اول آن را به شکل زنده بوت کنید تا مطمئن شوید که با سخت افزار شما مشکل سازگاری ندارد.

- آزمایش نرم‌افزارها: مثلا برای تست نسخه جدید مسنجر امپاتی، نیازی نیست حتما آن را نصب کنید بلکه می‌توانید توسط یک دیسک زنده که از این نسخه جدید استفاده می‌کند، آن را در حال کار ببینید و تست کنید.

ذکر این نکته هم مهم است که این روزها یو.اس.بی.های زنده هم وارد بازار شده‌اند. در این حالت به جای سی.دی. از یک درایو یو.اس.بی. (مثلا کول دیسک) استفاده می‌شود که درست مثل یک دیسک زنده قابلیت بوت کردن سیستم را دارد اما با یک مزیت مهم: قابل نوشتن است. وقتی از درایو یو.اس.بی. به جای سی دی استفاده می‌شود، تغییرات شما روی سیستم عامل قابل ذخیره شدن روی یو.اس.بی. خواهند بود و این می‌تواند برای کسانی که به شکل حرفه‌ای برای کار از دیسک‌های زنده استفاده می‌کنند، کمک بزرگی باشد.

بعد از اینکه استفاده، آزمایش یا هر چیز دیگری که باعث شده از دیسک زنده استفاده کنید به پایان رسید، کافی است از منو، ری استارت یا شات‌داون را انتخاب کنید و پس از پایان شات‌داون و بیرون آوردن سی دی از درایو، کامپیوتر را دوباره روشن کنید و همه چیز را در وضعیت قدیمی تحویل بگیرید.

۵.۲ قدم های مرسوم بعد از نصب لینوکس دسکتاپ

گاهی ممکن است یک سایت یا سند یا حتی برنامه به اجبار سعی در استفاده از یک فونت خاص داشته باشد. مثلا به این اسکرین شات نگاه کنید:

تیتر صفحه با صفحه کلید عربی (احتمالا در ویندوز) تایپ شده که به جای «ک» از «ك» و به جای «ی» از «ي» استفاده می‌کند. همچنین طراح به اجبار به صفحه گفته از فونت Tahoma استفاده کند که یک فونت ویندوزی است. برای درست دیدن این صفحه در مرحله اول باید از طراح و نویسنده خواست که از استانداردهای فارسی استفاده کنند و در مرحله دوم، برای حل موقتی مشکل (در اصل برای اضافه کردن امکان نمایش فارسی حروف عربی) باید فونت‌های استاندارد ویندوز را به سیستم اضافه کرد.

در اوبونتو بسته‌ای به نام `ubuntu-restricted-extras` که مجموعه ای است از ابزارهایی که به علت مجوزهای محدود کننده‌ای که شرکت‌های سازنده روی آن ها گذشته‌اند نمی‌توانند روی سی دی اصلی اوبونتو عرضه شوند. بگذارید نگاهی به محتویات آن بیندازیم:

```
jadi@jubung:~$ aptitude show ubuntu-restricted-extras
Package: ubuntu-restricted-extras
New: yes
State: installed
Automatically installed: no
Version: 57
Priority: optional
Section: multiverse/metapackages
Maintainer: Michael Vogt
<michael.vogt@ubuntu.com>
Architecture: amd64
Uncompressed Size: 30.7 k
Depends: ubuntu-restricted-addons
Recommends: ttf-mscorefonts-installer, unrar,
gstreamer0.10-plugins-bad-multiverse,
libavcodec-extra-53
Conflicts: ubuntu-restricted-extras
Description: Commonly used restricted packages for
Ubuntu
This package depends on some commonly used packages in
the Ubuntu multiverse repository.
Installing this package will pull in support for MP3
playback and decoding, support for various other audio
formats (GStreamer plugins), Microsoft fonts, Flash
plugin,
LAME (to create compressed audio files), and DVD
playback.
Please note that this does not install libdvdcss2, and
will not let you play encrypted DVDs.  For more
information, see
https://help.ubuntu.com/community/RestrictedFormats
/PlayingDVDs
Please also note that packages from multiverse are
restricted by copyright or legal issues in some
countries.  See http://www.ubuntu.comubuntu/licensing
for more information.
```

آنطور که این دستور می‌گوید، این بسته حاوی پشتیبانی از فرمت mp3 و فرمت‌های صوتی تصویری دیگر، پشتیبانی از فونت‌های مایکروسافت، فلش و پخش دی وی دی است. پس احتمالا با نصب آن، مشکل ما حل خواهد شد.

```
sudo apt-get install ubuntu-restricted-extras
```

به عنوان یک نکته اضافی ، خوب است بدانیم که هر کاربر گنو/لینوکس می‌تواند با ساخت پوشه‌ای به اسم .fonts در خانه خودش (home folder) و کپی کردن هر فونتی که لازم دارد درون آن و خروج و ورود مجدد به سیستم، به آن فونت‌ها دسترسی داشته باشد. مثلا کپی کردن فایل tahoma.ttf و tahomab.ttf می تواند مشکل وب سایت بالا و بقیه سایت‌هایی که از فونت تاهوما استفاده می‌کنند را حل کند.

توجه: هر فایل که در گنو/لینوکس و سیستم‌های یونیکسی که با نقطه شروع شود، در حالت عادی مخفی است و به کاربر نمایش داده نمی شود. در براوزر فایل گنوم (ناتیلوس) می‌توانید با فشردن Ctrl+H فایل‌های مخفی را ببینید.

۶.۲ نصب نرم افزارها

مفهوم نصب برنامه

مفهوم مدیر بسته

نصب با رابط گرافیکی مدیر بسته

نصب با مدیر بسته در خط فرمان

نصب از طریق کامپایل برنامه

نکته اصلی در این روش اینه که معمولا نباید استفاده بشه (: در این روش سیستم عامل و مدیر بسته، چیزی از برنامه‌های نصب شده نمی‌داند و نمی‌تواند آنها را بروزرسانی کند. در عین حال این روش کمی سخت‌تر از روش‌های قبله و گاهی هم نیاز به حوصله و دقت و چیز یاد گرفتن داره.

این روش را معمولا وقتی به کار می‌بریم که راه دیگری نباشد. این اتفاق در اکثر موارد مربوط است به نصب برنامه یا نسخه‌ای از آن که در منابع موجود نیستند.

برای شروع اول باید متن برنامه مورد نظر را دانلود کنیم. در این مورد، با مثال نصب «دیشکنری انگلیسی به فارسی سیب» پیش می‌روم. یک جستجو در اینترنت ما را به سایپ پروژه سیب می‌رساند. منطقا روی بخش `Download` کلیک می‌کنیم.

خب.. اگر دو کار سخت در کامپایل باشد، دومی این است که کدام فایل را باید دانلود کنیم. قدم به قدم!

از بخش قبلی یادمان هست که `RPM` پسوند بسته‌های ردهتی است. پس از این یکی می‌گذریم چون در حال حاضر اوبونتو داریم. اگر نسخه آر پی امی (مثل زوزه و ردهت و فدورا) داشتیم، احتمالا کار به سادگی دانلود و دوبار کلیک کردن روی این فایل بود.

در قدم بعدی دنبال فایل‌هایی با پسوند `tar.gz` می‌گردم. فایل تار نتیجه چسابندن چند فایل به هم و فایل جی.زد. نتیجه فشرده کردن فایل‌ها است. به عبارت دیگر فایلی با پسوند تار.جی.زد، مجموعه چند فایل خواهد بود که به هم فشرده شده‌اند.

بین نسخه‌های موجود، به سراغ بالاترین نسخه می‌رویم. به شرطی که در مرحله بتا و آلفا و این چیزها نباشد. اینجا ظاهرا نسخه ۸.۰ بهترین نسخه است. همان را دانلود می‌کنم.

حال فایل متنی برنامه را دارم. اول آن را از حالت فشرده خارج می‌کنم. در حالت گرافیکی به سادگی با کلیک راست و انتخاب `Extract` و در حالت متنی با دستوری شبیه به:

```
$ tar xf sib-0.8.tar.bz2
```

حالا یک دایرکتوری سیب دارم که داخلش می‌شوم و فایل‌هایش را بررسی می‌کنم.

```
jadi@jubun:~/Desktop$ cd sib-0.8/
jadi@jubun:~/Desktop/sib-0.8$ ls
AUTHORS ChangeLog COPYING makefile misc pixmaps README
src tools
```

تقریبا همیشه فایلی هست که شروع کار باشد. اینجا اسمش README است، ممکن است دیگران اسم‌هایی مثل read.me یا read.first یا هر چیز دیگری بگذارند ولی کامل امشخص است که از کجا باید شروع کرد. در انتهای این فایل به سادگی نوشته:

```
Basic Installation
==================
1. make
2. make install (root user)
```

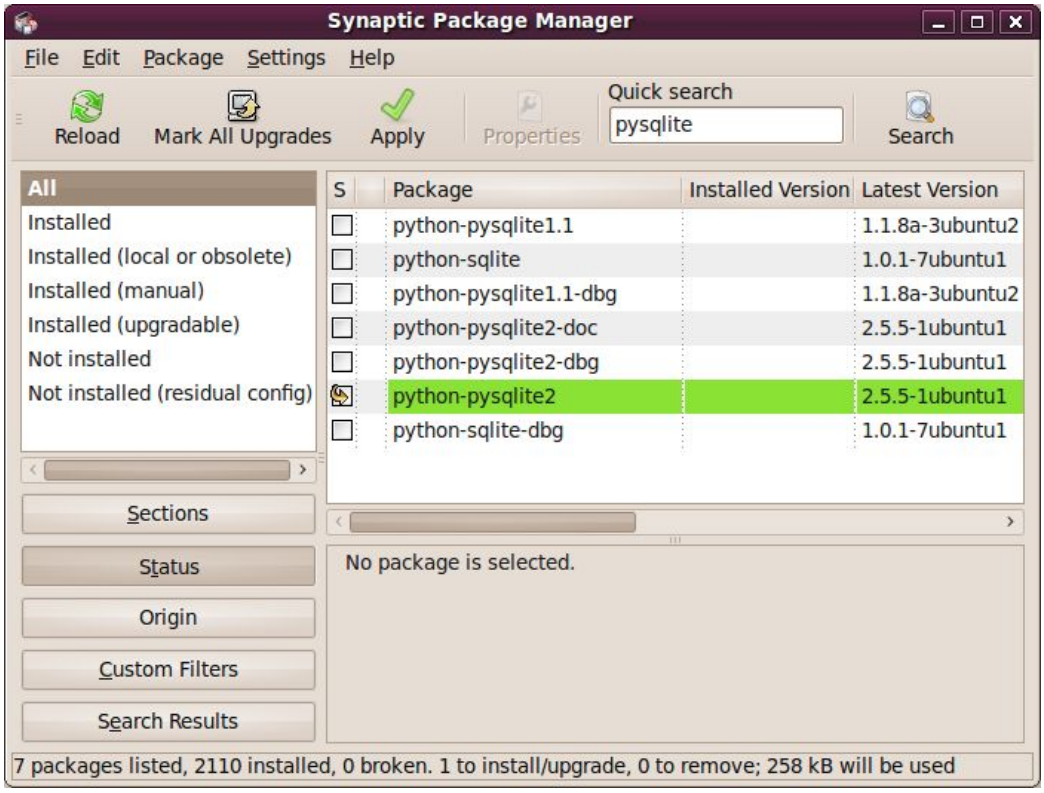
به همین سادگی! در این مورد حتی نیازی به قدم سخت configure/. هم نیست. در بسیاری از موارد اول باید دستور configure/. را تایپ کنید تا چک کند که سیستم شما دارای همه موارد مورد نیاز برای نصب یک برنامه باشد و بعد برنامه را کامپایل و بعد نصب کنید. سه قدم مشهور اینها هستند:

```
./configure
make
sudo make install
```

این سه دستور معمولا تمام برنامه‌ها را نصب می‌کنند مگر اینکه در راهنمای نصب برنامه، چیز دیگری نوشته شده باشد. البته باید توجه کنید که بخش کانفیگور معمولا در دفعات اول به Error هایی مبنی بر نبودن یکی دو کتابخانه یا برنامه لازم برای اجرای این برنامه می‌خورد که باید جدا جدا آن‌ها را پیدا و نصب کنید و دوباره کانفیگور را اجرا کند. بدون اجرای موفق کانفیگور، شانسی برای نصب نخواهید داشت (: بهترین روش هم جستجو است در مدیر نصب برنامه گرافیکی (مثلا Synaptic) به دنبال اسم برنامه‌هایی که کانفیگور از آن‌ها ایراد نصب گرفته است. متاسفانه این برنامه configure/. ندارد که دقیق به ما بگوید آیا همه نیازمندی‌های برنامه نصب هستند یا نه در نتیجه - به شکل منطقی و کاری که از اول باید می‌شد - به فایل README رجوع می‌کنم و می‌خوانم که برنامه به این‌ها نیاز دارد:

```
pygtk (>= 2.10)، python-sqlite2
python-notify (>=0.1).
```

حتی اگر نمی‌دانم اینها چیستند، Synaptic را باز می‌کنم و دنبالشان می‌گردم:



و نصبشان می‌کنم. اگر یادتان باشد، آن بالا گفتم که پیدا کردن اینکه دقیقا کدام فایل باید دانلود بشود، کار سخت دوم است. این پیدا کردن و نصب کتابخانه‌ها و وابستگی‌ها کار سخت اول است (: معمولا اسم‌ها دقیقا مشابه هم نیستند. مثلا در اینجا راهنما pygtk را لازم دارد اما در اوبونتوی من، اسم بسته python gtk است. همچنین توجه دارم که در synaptic ، دگمه جستجوی واقعی خیلی خیلی بهتر از آن جعبه جستجوی سریع بالای پنجره، کار می‌کند.

حالا که وابستگی‌ها نصب شده‌اند، طبق دستور README اول make را اجرا می‌کنم و

```
$ make
for d in src; do make -C $d; [ $? = 0 ] || exit 1 ; done
make[1]: Entering directory
/home/jadi/Desktop/sib-0.8/src’
Byte-compiling python modules...
bgl2sdb.py dbmanager.py dbquery.py prefrence.py sib.py
Byte-compiling python modules (optimised versions) ...
bgl2sdb.py dbmanager.py dbquery.py prefrence.py sib.py
make[1]: Leaving
directory/home/jadi/Desktop/sib-0.8/src’
```

و بعد برای نصب در سیستم، `make install` را با دسترسی مدیرسیستم (و برای اینکار یک `sudo` جلوی دستور می‌گذارم):

```
$ sudo make install
mkdir -p /usr/share/sib
mkdir -p /usr/share/pixmaps/sib
mkdir -p /usr/share/applications
mkdir -p /usr/share/gnome/autostart
mkdir -p /usr/bin
install -m644 COPYING /usr/share/sib/.
install -m644 pixmaps/*.png /usr/share/pixmaps/sib/.
install -m755 misc/sib /usr/bin/.
```

حالا احتمالا برنامه باید به درستی نصب شده باشد. آن را اجرا می‌کنم (:

```
$ sib
Traceback (most recent call last):
  File ”sib.py”, line 411, in
    main()
  File ”sib.py”, line 179, in init
    self.DB_con = dbquery.Database(DATADIR)
  File ”/usr/share/sib/dbquery.py”, line 51, in init
    self.list_of_db()
  File ”/usr/share/sib/dbquery.py”, line 61, in
list_of_db
    DBdir = os.listdir(self.home)
OSError: Errno 2 No such file or directory:
’/home/jadi/.sib/’
```

مثل اینکه درست اجرا نشد (: خطوط را می‌خوانم. این کاری است که نصب از سورس را کمی سخت کرده ولی فراموش نکنید که اگر مشغول نصب یک برنامه از فایل سورس هستید، یعنی در دنیای حرفه‌ای ها پا گذاشته‌اید و نباید از چیزی بترسید. خط آخر به وضوح نوشته که نمی‌تواند دایرکتوری `.sib` را در خانه من پیدا کند. این یک اشکال در روند نصب است. اول این ایراد را به نویسنده برنامه اطلاع می‌دهم و بعد خودم دستی یک دایرکتوری به نام `.sib` در خانه می‌سازم.

```
$ mkdir ∼/.sib
$ sib
```

و حالا سیب به خوبی اجرا می‌شود. مطمئنا از این به بعد دیگر آن را از منو اجرا خواهم کرد و نه از خط فرمان.

بگذارید یک نگاه دیگر به کل روند بیندازیم:

فایل را دریافت کردیم فایل را از حالت فشرده خارج کردیم در اینجا نه، اما در حالت معمول با `./configure` داشتن تمام نیازهای برنامه را بررسی کردیم آنقدر کتابخانه و برنامه جانبی نصب کردیم تا کانفیگور مطمئن شود که همه چیز مرتب است و با موفقیت اجرا بشود بعد دستور `make` را اجرا کردیم تا برنامه کامپایل شود و فایل اجرایی ساخته شود در نهایت `sudo make install` با دسترسی مدیر سیستم، فایل اجرایی و فایل‌های تنظیمات را در جاهایی که لازم بودند ساخت و حالا برنامه ما نصب شده است (: تنها سه نکته دیگر

را باید اضافه کنم:

نترسید. کامپایل برنامه یک کار حرفه‌ای است و شما مشغول یک کار جدی. با حوصله باشید و چیز یاد بگیرید. اکثر برنامه‌ها علاوه بر بسته معمولی نیاز به بسته‌هایی که در انتهای اسمشان `dev-` باشد هم دارند. اگر بسته‌ای را نصب می‌کنید و هنوز مشکل‌دارید، نسخه `dev-` را هم نصب کنید. اکثر سیستم‌ها خیلی از بسته‌های مورد نیاز برای کامپایل برنامه را در یک بسته بزرگ مجازی جمع کرده‌اند. ایده خوبی است که قبل از نصب و به جای جواب به تک تک وابستگی‌ها، اول آن را نصب کنید. در اوبونتو اسم این بسته مجازی `build-essential` است.

۷.۲ نرم‌افزارهای روزمره

اگر از پلتفرم ویندوز یا مکینتاش به لینوکس مهاجرت میکنید، شاید متعجب شوید اگر برنامه‌هایی که قبلاً استفاده میکردید برای لینوکس هم موجود باشند. بعضی از نرم‌افزارهایی که هم‌اکنون استفاده میکنید نسخه‌های لینوکسی دارند. برای آن دسته که به این شکل نیستند برنامه‌های آزاد و متن بازی موجود هست که نیازهای شما را پوشش می‌دهند. در این بخش نرم‌افزارهایی را معرفی می‌کنیم که به خوبی روی لینوکس کار می‌کنند[۱]:

- **Office Suites** (مجموعه‌ی آفیس)
  - Windows: Microsoft Office, LibreOffice
  - Apple os x: iWork, Microsoft Office, LibreOffice
  - Linux: LibreOffice, KOffice, gnome Office, Kexi (database application)
- **Email Applications** (برنامه‌های پست الکترونیکی)
  - Windows: Microsoft Outlook, Mozilla Thunderbird
  - Apple os x: Mail.app, Microsoft Outlook, Mozilla Thunderbird
  - Linux: Mozilla Thunderbird, Evolution, KMail
- **Web Browsers** (مرورگرهای وب)
  - Windows: Microsoft Internet Explorer, Mozilla Firefox, Opera, Chromium, Google Chrome
  - Apple os x: Safari, Mozilla Firefox, Opera, Chromium, Google Chrome
  - Linux: Mozilla Firefox, Opera, Chromium, Google Chrome, Epiphany
- **PDF Readers** (خواننده‌های پی‌دی‌اف)

[۱] منبع: ترجمه و تلخیص از کتاب Getting Started with Ubuntu 14.04 صفحات ۳۵-۳۷

- Windows: Adobe Acrobat Reader, Foxit
- Apple os x: Adobe Acrobat Reader
- Linux: Evince, Adobe Acrobat Reader, Okular

- **Multimedia Players** (پخش‌کننده‌های چندرسانه‌ای)
  - Windows: Windows Media Player, vlc
  - Apple os x: Quicktime, vlc
  - Linux: Totem, vlc, MPlayer, Kaffeine
- **Music Players and Podcatchers** (پخش‌کننده‌های موسیقی و پادکچرها)
  - Windows: Windows Media Player, iTunes, Winamp
  - Apple os x: iTunes
  - Linux: Rhythmbox, Banshee, Amarok, Audacity, Miro
- **CD/DVD Burning** (رایت سی‌دی/دی‌وی‌دی)
  - Windows: Nero Burning rom, InfraRecorder
  - Apple os x: Burn, Toast Titanium
  - Linux: Brasero, K3b, Gnome-baker
- **Photo Management** (مدیریت عکس)
  - Windows: Microsoft Office Picture Manager, Picasa
  - Apple os x: Aperture, Picasa
  - Linux: Shotwell, gThumb, Gwenview, F-Spot
- **Graphics Editors** (ویرایشگرهای گرافیکی)
  - Windows: Adobe Photoshop, gimp
  - Apple os x: Adobe Photoshop, gimp

- Linux: gimp, Inkscape

- **Instant Messaging** (مسنجر)
  - Windows: Windows Live Messenger, aim, Yahoo! Messenger, Google Talk
  - Apple os x: Windows Live Messenger, aim, Yahoo! Messenger, Adium, iChat
  - Linux: Empathy, Pidgin, Kopete

- **VoIP Applications** (برنامه‌های ویپ)
  - Windows: Skype, Google Video Chat
  - Apple os x: Skype, Google Video Chat
  - Linux: Ekiga, Skype, Google Video Chat

- **BitTorrent Clients** (کلاینت‌های بیت‌تورنت)
  - Windows: µTorrent, Vuze
  - Apple os x: Transmission, Vuze
  - Linux: Transmission, Deluge, KTorrent, Flush, Vuze, BitStorm Lite

فصل ۳

مباحث پیشرفته

۱.۳ ساختار فایل‌ها و دایرکتوری‌ها

امیدوارم حالا که به اینجای کتاب رسیده‌اید، با مفهوم فایل و دایرکتوری آشنا باشید. فایل‌ها مانند برگه‌های کاغذی هستن که روی آن‌ها اطلاعات مورد نظر چاپ شده و دایرکتوری‌ها مانند کشوها یا پوشه‌هایی که فایل‌ها در داخل آن‌ها قرار گرفته‌اند. در لینوکس این موضوع هم مانند بسیاری چیزهای دیگر از یک استاندارد پیروی می‌کند. استاندارد ناظر بر فایل‌ها و دایرکتوری‌ها **استاندارد سلسله مراتب فایل‌سیستم** یا به اختصار FHS نامیده می‌شود. نسخه ابتدایی این استاندارد در سال ۱۹۹۴ تنظیم شده بود و نسخه فعلی در سوم ژوئن ۲۰۱۵ منتشر شده است.

در استاندارد FHS ، تمام فایل‌ها و دایرکتوری‌های روی یک سیستم با در یک دایرکتوری ریشه که با / نشان داده می‌شود نگهداری شوند. حتی اضافه کردن یک دیسک سخت افزاری جدید یا متصل کردن یک هارد اکسترنال یا ماونت کردن[۱] یک دیسک روی شبکه هم تنها باعث اختصاص یک دایرکتوری به آن می‌شود. این کار ممکن است به صورت خودکار یا دستی انجام شود ولی در نهایت برای دسترسی به هر سخت مکان ذخیره فایل، کاربر باید وارد یکی از دایرکتوری‌هایی شود که جایی در زیر دایرکتوری ریشه ساخته شده است.

پس دیدیم که در لینوکس (که فرزند خلف یونیکس و استانداردهایش است) همه چیز تحت دایرکتوری / که به آن دایرکتوری ریشه یا Root Directory می گوییم قرار دارد. بگذارید نگاهی به دایرکتوریهای درون آن بیندازیم:

```
jadi@wonderland:~$ ls /
bin etc initrd.img.old lost+found opt run sys var
boot home lib media proc sbin tmp vmlinuz
dev initrd.img lib64 mnt root srv usr vmlinuz.old
```

البته بگذارید از دستور قشنگ‌تر tree استفاده کنم که ساختار درختی فایل‌ها را هم نشان می‌دهد. سوییچ L مشخص می‌کند که چند لایه تو در تو نمایش داده شود (در اینجا 1 ؛ یعنی فقط همین دایرکتوری) و سوییچ d درخواست می‌کند که فقط دایرکتوری‌ها به نمایش در بیایند و فایل‌ها نشان داده نشوند.

[۱] یعنی متصل کردن فضای جدید به سیستم و اختصاص یک دایرکتوری به آن

```
jadi@wonderland:~ $ tree / -L 1 -d
/
├── bin
├── boot
├── dev
├── etc
├── home
├── lib
├── lib64
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin
├── srv
├── sys
├── tmp
├── usr
└── var
```

این تقریبا همان چیزی است که استاندارد FHS مشخص کرده. هر سیستم عامل مبتنی بر یونیکس (از جمله لینوکس) حاوی دایرکتوری بندی مشابهی است و درک این دایرکتوری بندی یکی از اولین قدم‌های جدی برای تبدیل شدن به یک متخصص لینوکس است. چرا؟ چون دانستن اینکه هر چیزی در کجا قابل پیدا شدن است، باعث می‌شود شما حتی در مواردی که اطلاعات دقیقی هم ندارید، بتوانید قدم‌های اول را بردارید. مثلا اگر بدانید که همه تنظیمات در دایرکتوری `/etc` هستند و مدیرتان از شما بخواهد تنظیمات برنامه `vsftpd` را تغییر دهید، می‌توانید انتظار داشته باشید که این تنظیمات در در `/etc/vsftpd` یافت شوند یا اگر کسی گلایه کند که پروکسی‌اش کار نمی‌کنید تنها چیزی که برای شروع عملیات لازم دارید دانستن این است که همه لاگ‌های سیستم در `/var/log/` هستند و با رفتن به آن‌جا می‌توانید به دنبال ریشه مشکلات بگردید.

پس بگذارید نگاهی دقیق‌تر به دایرکتوری‌های بالا بیندازیم و ببینیم در ریشه یک سیستم لینوکس، چه چیزهایی یافت می‌شوند.

دایرکتوری ها و کاربرد آنها

۱. **/**: همانطور که گفته شد، این دایرکتوری ریشه است که همه فایل‌ها و دایرکتوری ها (و دستگاه های متصل به سیستم که به شکل یک فایل یا دایرکتوری شناخته می‌شوند) در آن قرار دارند

۲. **/bin**: فایل‌های اجرایی مهم در اینجا هستند. بخصوص دستوراتی که باید حتی در حالت تک کاربره و مشکل‌یابی نیز در دسترس باشند یا آن‌هایی که توسط همه کاربران مورد استفاده قرار می‌گیرند؛ چیزهایی مثل دستور `cat` یا `ls` یا `cp`

۳. **/boot**: اینجا مجموعه فایل‌های مرتبط با بوت لودر قرار گرفته‌اند که وظیفه بوت اولیه سیستم را بر عهده دارند. چیزهایی مانند کرنل و `initrd` و بوت کننده `.grub`

۴. **/dev**: ابزارهای اساسی اینجا قرار گرفته‌اند. در لینوکس همه چیز یا پروسه است یا فایل و همین فلسفه باعث شده که تک تک ابزارهای سخت افزاری متصل به سیستم هم در این دایرکتوری به شکل یک فایل شناخته شوند. برای مثال `/dev/video0` می‌تواند نشان دهنده وبکم و `/dev/random` ابزاری باشد که هربار از آن بخوانید، یک عدد اتفاقی به شما تحویل می‌دهد.

۵. **/etc**: این دایرکتوری یکی از اصلی‌ترین مکان‌های مورد مراجعه هر لینوکس کار است. کلیه تنظیمات سیستمی در اینجا قرار می‌گیرند. معمولا هر برنامه در این دایرکتوری برای خود یک دایرکتوری می سازد و تنظیمات مورد پذیرش در کل سیستم را آنجا می‌گذارد. در برخی موارد هم که برنامه‌ها کوچکتر هستند و تنها یکی دو فایل تنظیم دارند، فقط با استفاده از اسم خود و بدون ساخت زیرشاخه، تنظیمات را در این مکان ذخیره می‌کنند. به عنوان یک نمونه، `/etc/X11` حاوی تنظیمات مربوط به محیط گرافیکی (که `X11` نامید می‌شود) است. این دایرکتوری مهم همچنین شامل تنظیماتی است که مربوط به برنامه‌هایی می‌شود که در هر مرحله از بوت شدن سیستم باید اجرا شوند و تنظیماتی که به شکل دوره‌ای برنامه‌هایی را اجرا می‌کنند (کرون ها). در مستندات اولیه یونیکس آزمایشگاه‌های بل، کلمه `etc` به عنوان مخفف `etcetera` عنوان شده که به معنی **متفرقه** است و نشان دهنده آنکه برنامه‌های متفرقه (به جز خود لینوکس) تنظیمات خود را در اینجا ذخیره می‌کنند. مستندات جدیدتر گاهی از آن با عنوان تنظیمات قابل ویرایش (*`Editable Text Configuration`*) یا گنجه ابزارهای اضافی (*`Extended Tool Chest`*) هم نام می‌برند که مخفف با معناتری است.

۶. **/home**: دایرکتوری شخصی کاربران در اینجا قرار دارد. برای مثال اگر شما در سیستم کاربری به نام `jadi` بسازید، دایرکتوری `/home/jadi` به فایل‌های شخصی، تنظیمات فردی و چیزهای

دیگری اختصاص می‌یابد که این کاربر بر روی سیستم ذخیره می‌کند. هر کاربر در خانه شخصی خود حق نوشتن، خواندن و اجرای فایل‌ها را دارد.

۷. **/lib**: کتابخانه‌های برنامه نویسی که توسط فایل‌های اجرایی مانند آن‌هایی که در `bin` یا `sbin` هستند در اینجا قرار می‌گیرد.

۸. **/media**: دایرکتوری‌ای موقت برای ماونت کردن ابزارهای جانبی. مثلا اگر شما `jadi` باشید و یک دیسک یو اس بی به نام `myusbdisk` به کامپیوتر لینوکس دسکتاپ خود متصل کنید، آن را در مسیر `/media/jadi/myusbdisk` خواهید یافت. این مسیر برای بقیه ابزارهای ذخیره سازی جداشونده از جمله سی‌دی درایوها نیز استفاده می‌شود.

۹. **/mnt**: مسیری است عمومی برای ماونت کردن ابزارهای ذخیره سازی. این مسیر بر خلاف مسیر `/media` معمولا تنها توسط مدیرسیستم یا روت برای متصل کردن ابزارهای جانبی استفاده می‌شود.

۱۰. **/opt**: این مسیر معمولا برای نصب برنامه‌های وندورهای مستقل استفاده می‌شود. برای مثال در صورتی که برنامه‌ای مانند اوراکل را خریداری کنید، برنامه در مسیر `/opt` که مخفف کلمات `optional` است نصب خواهد شد.

۱۱. **/proc**: تکرار می‌کنم که همه چیز در لینوکس یا فایل است یا پروسه و اضافه می‌کنم که این دایرکتوری برخورد این دو با یکدیگر است. در این دایرکتوری که عملا یک فایل سیستم مجازی است، هر پروسه در حال اجرا در سیستم و اطلاعاتی مربوط به وضعیت سیستم به شکل یک فایل به نمایش درمی آید.

۱۲. **/root**: این دایرکتوری خانه کاربر روت است و فایل‌های شخصی یا تنظیمات اختصاصی کاربر روت در این مکان ذخیره می‌شود. توجه نمایید که علی رغم شباهت اسمی، این دایرکتوری هیچ ربطی به `/` که دایرکتوری روت خوانده می‌شود ندارد.

۱۳. **/run**: اطلاعاتی مربوط به سیستم از زمان آخرین بوت. چیزهایی مانند کاربران وارد شده به سیستم و دامون‌های در حال اجرا در اینجا قابل دسترسی هستند. توجه داشته باشید که اینبار لازم ندیدم تذکر بدهم که همه چیز یا فایل است یا پروسه و این دایرکتوری نشان دهنده پروسه‌ها، به شکل فایل.

۱۴. **/sbin**: فایل‌های اجرایی حیاتی در اینجا قرار می‌گیرند. چیزهایی مانند `init` و `mount.`

۱۵. **/tmp**: فایل‌های موقتی که برنامه‌ها یا کاربران آن‌ها را ساخته‌اند. از این دایرکتوری نباید انتظار امنیت یا حتی پایداری داشت. همه کاربران به این دایرکتوری دسترسی دارند و ممکن است بعد از بوت کاملا پاک شود.

۱۶. **/usr**: این دایرکتوری و دایرکتوری‌های درون آن حاوی کتابخانه‌های برنامه نویسی، اسناد، برنامه‌ها و حتی سورس برنامه‌هایی هستند که در سیستم نصب شده. زیردایرکتوری‌های این شاخه، معمولا به شکل زیر مرتب شده‌اند:

(آ) **/usr/bin**: برنامه‌های غیر ضروری سیستم مانند ابزارهایی مورد استفاده کاربران عمومی. چیزهایی مانند آفیس یا مرورگر وب

(ب) **/usr/include**: فایل‌های هدر برنامه‌نویسی

(ج) **/usr/lib**: کتابخانه‌های برنامه نویسی مورد استفاده عموم کاربران

(د) **/usr/local**: برنامه‌های سیستمی غیرضروری. چیزهایی مثل سرویس‌های شبکه یا پرینترها اینجا قرار می‌گیرند.

(ه) **/usr/src**: کد متن برنامه‌ها در صورت نصب شدن در اینجا قرار می‌گیرند.

(و) **/usr/X11R6**: بالاتر گفتیم که X11 لایه پایینی سیستم گرافیکی لینوکس است. فایل‌های اجرایی آن در اینجا قرار دارند.

۱۷. **/var**: این دایرکتوری همانطور که از نام آن یعنی variable می‌شود حدس زد، حاوی فایل‌هایی است که انتظار می‌رود دائما در طول کارکرد مرسوم سیستم تغییر کنند. چیزهایی مانند لاگ‌های سیستم، فایل‌های سرور، ایمیل‌های در حال ارسال و اسناد منتظر پرینت شدن و دیتابیس‌ها. این دایرکتوری معمولا حاوی زیرشاخه‌هایی به شکل زیر است:

(آ) **/var/cache**: در صورتی که نرم‌افزاری نیاز به ذخیره موقت به شکل کش داشته باشد، در این مسیر برای خودش یک دایرکتوری می‌سازد. انتظار می‌رود که همیشه فایل‌های کش بدون صدمه زدن به سیستم قابل پاک کردن باشند.

(ب) **/var/lib**: فایل‌های نشان دهنده وضعیت اینجا هستند. اطلاعاتی که توسط برنامه‌ها تغییر می‌کنند ولی باید همیشه در دسترس بمانند. چیزهایی مانند دیتابیس‌ها، اطلاعات مربوط به مدیر بسته و غیره.

(ج) **/var/lock**: فایل‌های قفل. اینها ساخته می‌شوند تا یک پروسه بداند که آیا پروسه مشابهی در این لحظه در حال اجرا است یا نه. برای مثال در صورتی که مدیر بسته `apt` را اجرا کنید در طول زمان اجرایش، فایلی در اینجا می‌سازد تا به بقیه مدیربسته‌هایی که ممکن است اجرا شوند، حضور خود را یادآوری کند.

(د) **/var/mail**: ایمیل‌های داخلی کاربران در اینجا قرار می‌گیرند. توجه کنید که این ایمیل معمولا با ایمیل مرسومی که شما روی یکی از سرویس دهنده ها دارید نامرتبط است.

(ه) **/var/opt**: اطلاعات تغییر کننده ای که توسط برنامه‌های وندورهای دیگر (مثلا اوراکل) ایجاد می‌شوند.

(و) **/var/run**: اطلاعاتی در مورد سیستم از زمان بوت تا به حال به همراه اطلاعاتی در مورد کاربران وارد شده و دامون‌های در حال اجرا. بله! این دایرکتوری کپی `/run` است! یک کپی واقعی! در بسیاری سیستم‌ها این دو دایرکتوری عملا یک چیز هستند و فقط تصویری از یکدیگر.

(ز) **/var/spool**: سطلی از کارهایی که منتظرند کسی آن‌ها را بردارد. برای مثال وقتی دستور چاپ یک صفحه را می‌دهید، برنامه درخواست چاپ را در این دایرکتوری می‌گذارد تا سیستم چاپ سر فرصت آن را بردارد و چاپ کند. همین اتفاق برای ارسال ایمیل نیز می‌افتد. در صورت درخواست ارسال یک ایمیل داخلی، فایلی در اینجا ساخته می‌شود و برنامه ارسال ایمیل که دائما در حال نگاه به این دایرکتوری است، آن را برداشته، ارسال می‌کند.

(ح) **/var/tmp**: فایل‌های موقتی که کاربران یا برنامه‌ها ایجاد می‌کنند. تفاوت این دایرکتوری با `/tmp` این است که این دایرکتوری بین دو ریبوت سیستم پاک نمی‌شود در حالی که `/tmp` در بسیاری سیستم‌ها در هنگام بوت، خالی می‌شود.

نگران نباشید! دوبار مرور منطقی فهرست و درک دلایل نامگذاری دایرکتوری‌ها به راحتی باعث خواهد شد موضوع در ذهن شما بماند. البته اگر واقعا فکر می کنید نمی‌توانید همه جریان را در خاطر داشته باشید، فقط یادتان نگهداری که تنظیمات در `etc` هستند و لاگ‌ها در `/var/log` و `tmp` هم همیشه می‌تواند جای کار موقتی باشد که به زودی پاک خواهد شد.

منابع برای مطالعه بیشتر

برای اطلاعات کامل در مورد استاندارد FHS به ویکیپدیا رجوع کنید[۱]

[۱] `en.wikipedia.org`

۲.۳ دستورات معمول خط فرمان

خط فرمان لینوکس به شما اجازه می‌دهد دستوراتی را از صفحه کلید اجرا کنید و به فایل‌ها دسترسی داشته باشید. تعداد دستورات لینوکس بسیار زیاد است و هر دستور هم تنظیمات بسیار متنوعی دارد اما آن چیزهایی که معمولا استفاده می‌شوند را می‌شود در یکی دو درس گنجاند. فراموش نکنید که همانطور که در درس قابل خوانده‌اید، استفاده از خط فرمان چیزی است شبیه به بالا زدن کاپوت ماشین که هم به ما قدرت بیشتری در درک ساز و کار خودرو می‌دهد و هم ممکن است باعث خراب کردن موتور شود. اما شجاع باشید و به خودتان یادآوری کنید که انسان با تلاش و احیانا اشتباه کردن چیز یاد می گیرد و دنیای لینوکس دنیایی بسیار مهربان است و پر از آدم‌هایی که دوست دارند اشتباهات شما را با خوشحالی اصلاح کنند. این دنیا و اشتباه کردن در آن می‌تواند دریچه‌ای باشد برای درک اینکه بدون اشتباه کردن نمی‌شود چیزی را یاد گرفت پس همیشه یادتان نگه‌دارید که اگر کسی در مورد شما اشتباهی انجام داد، یعنی دارد تلاش می‌کند در مورد شما چیزی یاد بگیرد. حالا بگذارید نگاهی به معمول‌ترین دستورات خط فرمان بیندازیم.

کار با فایل‌ها و دایرکتوری‌ها

`ls` برای دیدن فهرست دایرکتوری‌ها و فایل‌ها

در هر جایی که باشید، با زدن دستور `ls` می توانید ببینید چه فایل‌ها و دایرکتوری‌هایی در آنجا وجود دارد. برای مثال:

```
jadi@wonderland:~$ ls
1.txt  digits  donot.read.me  empty  mystery  project
```

همانطور که می‌بینید خواندن این خروجی کمی سخت است و اطلاعات چندانی هم منتقل نمی‌کند. پس بهتر است از سوییچ‌ها استفاده کنیم. سوییچ‌های دستورات لینوکس معمولا بعد از `-` یا `--` نوشته می‌شوند. مثلا برای دیدن خروجی طولانی (`long`) باید از سوییچ `l` استفاده کنم:

```
jadi@wonderland:~$ ls -l
total 20
-rw-rw-r-- 1 jadi jadi   64 Jul  6 15:50 1.txt
-rw-rw-r-- 1 jadi jadi   21 Jul  6 15:50 digits
-rw-rw-r-- 1 jadi jadi   17 Jul  6 15:50 donot.read.me
-rw-rw-r-- 1 jadi jadi    0 Jul  6 15:50 empty
drwxrwxr-x 2 jadi jadi 4096 Jul  6 15:51 mystery
drwxrwxr-x 2 jadi jadi 4096 Jul  6 15:51 project
```

اطلاعات بیشتری در مورد هر فایل و دایرکتوری پیدا می‌کنیم. مثلا حجم هر فایل، تاریخ ساخته شدن آن و مالک و دسترسی‌ها (در این مورد بعدا بیشتر توضیح خواهیم داد). همچنین حرف `d` در اول خط نمایشگر این است که این خط مربوط به یک دایرکتوری است. ترکیب سوییچی بسیار مرسوم در گرفتن دایرکتوری `ltrh` است؛ `l` برای نمایش طولانی، `h` برای نمایش آدم‌وار (اول کلمه `human` که باعث می‌شود مثلا به جای حجم `4096` با عبارت آدم‌وار `4k` روبرو شویم) و `tr` برای تنظیم بر اساس «معکوس زمان» یا همان `time reversed` که جدیدترین فایل‌ها را پایین نمایش می‌دهد. چرا این فایل مهم است؟ چون می‌تواند به شکلی آدم‌وار به شما بگوید که آخرین فایل‌هایی که در یک دایرکتوری تغییر کرده‌اند چه بوده‌اند. مثلا در بسیاری مواقع از این برای کشف اینکه آیا لاگی تغییر کرده یا نه از چنین دستوری استفاده می‌کنیم:

```
jadi@wonderland:~$ ls -ltrh /var/log
total 1.5M
drwxr-xr-x 2 root      root 4.0K Apr 12 09:03 dist-upgrade
-rw-rw---- 1 root      utmp    0 Apr 17 01:32 btmp
-rw-r----- 1 root      adm    59 Apr 17 01:32 dmesg.1.gz
-rw-r--r-- 1 root      root  61K Apr 17 01:33 bootstrap.log
drwxr-xr-x 2 root      root 4.0K Jul  5 09:16 fsck
drwxr-xr-x 2 root      root 4.0K Jul  5 09:17 apt
drwxr-xr-x 3 root      root 4.0K Jul  5 09:32 installer
drwxr-xr-x 2 landscape root 4.0K Jul  5 09:32 landscape
-rw-r----- 1 root      adm   86K Jul  5 09:32 dmesg.0
-rw-r--r-- 1 root      root  32K Jul  5 09:53 faillog
-rw-r--r-- 1 root      root  20K Jul  5 10:01 alternatives.log
drwxr-xr-x 2 root      root 4.0K Jul  5 11:10 upstart
-rw-r--r-- 1 root      root 236K Jul  5 11:10 udev
-rw-r----- 1 root      adm   86K Jul  5 11:10 dmesg
-rw-r--r-- 1 root      root  179 Jul  5 11:10 boot.log
-rw-r--r-- 1 root      root  12K Jul  5 17:17 aptitude
drwxr-xr-x 2 root      root 4.0K Jul  6 06:42 unattended-upgrades
-rw-r----- 1 syslog    adm  256K Jul  6 06:42 kern.log
-rw-r--r-- 1 root      root 375K Jul  6 06:43 dpkg.log
-rw-rw-r-- 1 root      utmp  14K Jul  6 15:47 wtmp
```

```
-rw-rw-r-- 1 root      utmp 286K Jul  6 15:47 lastlog
-rw-r----- 1 syslog    adm   16K Jul  6 15:49 auth.log
-rw-r----- 1 syslog    adm  271K Jul  6 15:56 syslog
```

توجه. سوییچ‌ها را در طول زمان حفظ خواهید شد! نگران نباشید.

pwd برای دیدن مسیر جاری

دستوری بسیار ساده که به شما می‌گوید در کجای جهان ایستاده‌اید:

```
jadi@wonderland:/tmp$ pwd
/tmp
```

اکثر توزیع‌ها از جمله اوبونتویی که در این درس استفاده شده، در خط فرمان مسیری که در آن هستید را به شما نشان می‌دهند اما دستور pwd هم راه حل خوبی است برای اینکه به شما بگوید در حال حاضر در کدام دایرکتوری هستید. در لینوکس شما با دستوراتی مانند cd که چند لحظه دیگر آن را خواهیم دید می‌توانید در دایرکتوری ها حرکت کنید و pwd مانند یک جی.پی.اس. عالی در هر لحظه می‌تواند جای شما را به شما گزارش دهد. همانطور که قبلا هم گفته بودم تمام فایل‌های لینوکس در یک ساختار درختی از دایرکتوری‌ها چیده شده‌اند و شما در هر لحظه در یکی از این شاخه‌ها ایستاده‌اید. یادتان هست که شاخه‌ها از / که به آن روت می گفتیم شروع می شوند و تا هر کجایی که شما دایرکتوری‌های تو در تو بسازید ادامه پیدا می‌کنند.

cd برای حرکت در دایرکتوری ها

در بخش ساختار فایل‌های لینوکس یادگرفتیم که در لینوکس همه فایل‌ها در یک ساختار درختی از دایرکتوری‌ها قرار دارند که از دایرکتوری ریشه که آن را با / نشان می‌دهیم شروع شده و به سمت پایین ادامه می‌یابد. برای حرکت در این دایرکتوری‌ها از دستور cd که مخفف change directory است استفاده می‌کنیم. در جلوی این دستور باید مسیر جایی که می‌خواهیم به آنجا برویم را مشخص کنیم:

```
jadi@wonderland:~$ cd /
jadi@wonderland:/$
```

همانطور که می‌بینید کاربر در ابتدا در خانه خود بوده (که همیشه آن را با ~ نمایش می‌دهیم) و با زدن / در جلوی دستور سی دی، به دایرکتوری ریشه رفته است. در مرحله بعد این امکان را داریم که وارد یکی دیگر از دایرکتوری‌ها شویم:

```
jadi@wonderland:/$ cd log
-bash: cd: log: No such file or directory
```

```
jadi@wonderland:/$ cd var
jadi@wonderland:/var$ cd log
jadi@wonderland:/var/log$ ls
alternatives.log  aptitude  boot.log       btmp
dmesg    dmesg.1.gz  faillog  installer  landscape
syslog  unattended-upgrades  wtmp
apt               auth.log  bootstrap.log  dist-upgrade
dmesg.0  dpkg.log    fsck     kern.log   lastlog
udev    upstart
jadi@wonderland:/var/log$ cd apt
jadi@wonderland:/var/log/apt$ ls
history.log  term.log
jadi@wonderland:/var/log/apt$
```

همانطور که می بینید اول تلاش شده به دایرکتوری `log` وارد شویم و سیستم پاسخ داده که چنین فایل یا دایرکتوری‌ای موجود نیست . سپس ابتدا به دایرکتوری `var` رفته‌ایم و بعد با گرفتن `ls`، وارد دایرکتوری `apt` شده‌ایم و از آنجا `ls` گرفته‌ایم. این امکان از ابتدا موجود بود که با دستور `cd log/var/apt` وارد همان دایرکتوری شویم.

تفاوت مسیرهای محلی و مسیرهای کامل: در صورتی که در ابتدای هر مسیری / بگذاریم، داریم به دستور مورد نظر می‌گوییم که مسیر فایل را از دایرکتوری ریشه مشخص کرده‌ایم. مثلا در هرجایی از سیستم که باشیم می‌توانیم با موفقیت دستور `cd /var/log/apt` را اجرا کنیم. اما اگر ابتدای مسیردهی خود را با / آغاز نکنیم، مشغول دادن مسیرهای محلی هستیم یعنی با دستور `cd var/log/apt` به سیستم فرمان داده‌ایم که در همین جایی که هستم دنبال دایرکتوری *var* بگرد و داخل آن به دایرکتوری *log* برو و در نهایت به *apt* وارد بشو . این دستور در یک سیستم لینوکس معمول که دایرکتوری های اضافی در آن ایجاد نکرده باشیم، فقط از مسیر ریشه کار خواهد کرد چون فقط اگر در ریشه ایستاده باشیم، زیر پایمان یک دایرکتوری `var` وجود دارد.

درک شاخه‌بندی و جایی که ایستاده‌اید و مسیرهای محلی و مطلق در ابتدا کمی پیچیده است ولی در زندگی روزمره به سرعت آن را یاد خواهید گرفت. این نکته را هم در یاد داشته باشید که هر دایرکتوری حاوی دو فایل مجازی خاص هم هست: نقطه و دو نقطه. فایل نقطه به معنای همین دایرکتوری بوده و فایل دو نقطه به معنی یک دایرکتوری بالاتر است. منطقی است که زدن سی دی دو نقطه، باعث یک پله بالارفتن در درخت دایرکتوری‌ها شود:

```
jadi@wonderland:/var/log/apt$ cd ..
jadi@wonderland:/var/log$
```

و **آخرین نکته** ! در لینوکس هرچقدر هم که نوشیده باشید، رسیدن به خانه ساده است. زدن یک `cd` خالی، شما را از هر کجا به خانه‌تان خواهد رساند:

```
jadi@wonderland:/var/log$ cd
jadi@wonderland:~$ pwd
/home/jadi
```

این دستور معادل `cd ~` است چون ~ برای هر کاربر به معنی دایرکتوری خانه من است.

`cp` برای کپی کردن فایل‌ها

با دستور `cp` که مخفف کپی است، می‌توانید کپی جدیدی از فایل‌های موجود در سیستم تهیه کنید. دستور کلی به این شکل است:

```
cp file1 file2
```

که باعث کپی شدن فایل اول به اسم فایل دوم می‌شود (در نهایت دو نسخه از فایل خواهید داشت). دقت کنید که مانند تمام دستورات دیگر، فایل‌ها می‌توانند با نام مسیر همراه شوند:

```
jadi@wonderland:~$ ls
1.txt  digits  donot.read.me  empty  mystery  project
jadi@wonderland:~$ cp 1.txt 2.txt
jadi@wonderland:~$ ls
1.txt  2.txt  digits  donot.read.me  empty  mystery
project
jadi@wonderland:~$
```

می‌بینید که یک کپی از فایل `1.txt` گرفته‌ایم. همینکار را می‌شد با دادن مسیر نیز انجام داد:

```
jadi@wonderland:~$ ls
1.txt  2.txt  digits  donot.read.me  empty  mystery
project
jadi@wonderland:~$ ls mystery/
jadi@wonderland:~$ cp 1.txt mystery/
```

```
jadi@wonderland:~$ ls mystery/
1.txt
```

توجه کنید که چطور با زدن مسیر مورد نظر در جلوی دستور لیست، لیست فایل‌های درون آن دایرکتوری را بررسی کرده‌ایم.

rm برای حذف فایل‌ها

دستور rm که خلاصه remove است، فایل‌ها را حذف می‌کند، ساده و سر راست و خطرناک!

```
jadi@wonderland:~$ ls
1.txt  2.txt  digits  donot.read.me  empty  mystery
project
jadi@wonderland:~$ rm 2.txt
jadi@wonderland:~$ ls
1.txt  digits  donot.read.me  empty  mystery  projectt
```

مشخص است که زدن دستوری مانند rm * کل فایل‌های دایرکتوری موجود را حذف کرد. سوییچ مرسوم این دستور r- است که باعث می‌شود فایل‌ها به شکل recursive یا بازگشتی حذف شوند که در دنیای کامپیوتر به معنای این جا و همه دایرکتوری‌های توی اینجا است. در بررسی این دستور کمی احتیاط کنید ولی نگران هم نباشید و در زندگی هم فراموش نکنید که چیزهایی هست که بهتر است حذف شوند. یک rm ساده روی فایل‌هایی که آن‌ها را نمی‌خواهید می‌تواند فرصت‌های جدیدی برایتان فراهم کند.

mv برای تغییر نام یا مسیر فایل‌ها

این دستور که خلاصه move است، می‌تواند تقریبا مانند کپی، فایلی را از یک نام یا مسیر به نام یا مسیر دیگر انتقال دهد. این دستور بر خلاف دستور کپی، فایل اول را حذف و فایل دوم را ایجاد می‌کند.

```
jadi@wonderland:~$ ls
1.txt  digits  donot.read.me  empty  mystery  project
jadi@wonderland:~$ mv 1.txt 2.txt
jadi@wonderland:~$ ls
2.txt  digits  donot.read.me  empty  mystery  project
```

touch برای ایجاد یا آپدیت تاریخ یک فایل

این دستور در ساده‌ترین حالت یک نام فایل در جلوی خودش می‌گیرد و اگر فایلی به این اسم موجود نباشد، یک فایل خالی به آن نام می سازد اگر فایل از قبل موجود باشد، تاریخ آن را به تاریخ زمان حال آپدیت می‌کند با اینکه این فایل در این لحظه چندان مفید به نظر نمی رسد، از تعداد دفعاتی که در آینده آن را استفاده می‌کنید تعجب خواهید کرد.

```
jadi@wonderland:~$ ls -ltrh
total 20K
-rw-rw-r-- 1 jadi jadi   64 Jul  6 15:50 2.txt
-rw-rw-r-- 1 jadi jadi    0 Jul  6 15:50 empty
-rw-rw-r-- 1 jadi jadi   17 Jul  6 15:50 donot.read.me
-rw-rw-r-- 1 jadi jadi   21 Jul  6 15:50 digits
drwxrwxr-x 2 jadi jadi 4.0K Jul  6 15:51 project
drwxrwxr-x 2 jadi jadi 4.0K Jul  6 16:44 mystery
jadi@wonderland:~$ touch newfile
jadi@wonderland:~$ ls
2.txt  digits  donot.read.me  empty  mystery  newfile
project
jadi@wonderland:~$ touch 2.txt
jadi@wonderland:~$ ls -ltrh
total 20K
-rw-rw-r-- 1 jadi jadi    0 Jul  6 15:50 empty
-rw-rw-r-- 1 jadi jadi   17 Jul  6 15:50 donot.read.me
-rw-rw-r-- 1 jadi jadi   21 Jul  6 15:50 digits
drwxrwxr-x 2 jadi jadi 4.0K Jul  6 15:51 project
drwxrwxr-x 2 jadi jadi 4.0K Jul  6 16:44 mystery
-rw-rw-r-- 1 jadi jadi    0 Jul  6 16:52 newfile
-rw-rw-r-- 1 jadi jadi   64 Jul  6 16:52 2.txt
jadi@wonderland:~$
```

mkdir

با این دستور می‌توانید یک دایرکتوری بسازید:

```
jadi@wonderland:~$ ls
1.txt  digits  donot.read.me  empty  mystery  project
jadi@wonderland:~$ mkdir newdir
jadi@wonderland:~$ ls
1.txt  digits  donot.read.me  empty  mystery  newdir
project
jadi@wonderland:~$ mkdir newdir/newer
jadi@wonderland:~$ ls newdir/ -ltrh
total 4.0K
drwxrwxr-x 2 jadi jadi 4.0K Jul  6 16:54 newer
```

rmdir

این دستور یک دایرکتوری خالی را پاک می‌کند. مهم است که پیش از پاک کردن یک دایرکتوری فایل های درون آن را پاک کرده باشید[۱] .

```
jadi@wonderland:~$ ls
1.txt  digits  donot.read.me  empty  mystery  newdir
project
jadi@wonderland:~$ touch newdir/newfile
jadi@wonderland:~$ ls newdir/
newer  newfile
jadi@wonderland:~$ rmdir newdir/newer
jadi@wonderland:~$ ls newdir/
newfile
jadi@wonderland:~$ rmdir newdir/
rmdir: failed to remove ‘newdir’/: Directory not empty
jadi@wonderland:~$ rm newdir/newfile
```

[۱] یادتان هست؟ با دستور rm

```
jadi@wonderland:~$ ls newdir/
jadi@wonderland:~$ rmdir newdir/
jadi@wonderland:~$ ls
1.txt  digits  donot.read.me  empty  mystery  project
```

`cat` برای نمایش محتویات یک فایل متنی

دستور کت می‌توانید محتویات یک فایل متنی را نمایش دهد:

```
jadi@wonderland:~$ ls
1.txt  digits  donot.read.me  empty  mystery  project
jadi@wonderland:~$ cat digits
1
2
3
4
5
6
7
8
9
0

jadi@wonderland:~$ cat 1.txt
This is a normal text file
it has 3 lines
this is the last line
```

`less` برای بررسی محتویات فایل‌های متنی

این دستور برای دیدن محتوای یک فایل متنی استفاده می‌شود. با زدن `less filename` محتویات فایل به نمایش درمی‌آیند و شما می توانید با زدن کلیدهای بالا و پایین خط به خط در آن حرکت کنید یا با زدن

space یک صفحه به جلو بروید. برای خروج از برنامه less کافی است کلید q را فشار دهید و در صورتی که نیاز داشتید چیزی را در بین متن جستجو کنید، بعد از زدن یک / متن مورد جستجو را تایپ کرده، انتر را فشار دهید. زدن اسلش مجدد متن مورد جستجو را دوباره در بخش‌های جلوتر متن سرچ خواهید کرد و اگر خواستید از جایی که هستید رو به عقب جستجو کنید، کافی است دوباره ؟ را تایپ کنید و انتر بزنید. این را هم بگویم که رفتن به سر یک خط مشخص از طریق تایپ کردن شماره خط و بعد فشار دادن کلید g ممکن و است در نهایت G (g بزرگ) هم شماره به آخرین متن باز شده خط می‌برد. گیج شدن ممنوع ! به همین سادگی:

- / و بعد یک متن ، آن متن را جستجو می‌کند
- ? رو به عقب جستجو می‌کند
- بالا پایین و اسپیس در متن جلو عقب می‌رود
- شماره خط و g به آن خط خاص می‌رود
- G به آخرین خط می‌رود
- و q که اول quit است از لس خارج می‌شود.

بعضی کاربران قدیمی سیستم عامل فسیل شده داس[1] که بعدا به لینوکس مهاجرت کرده‌اند، بنا به عادت داسی قدیم خود از دستور more به جای less استفاده می‌کنند! هرگز اینکار را نکنید. less بسیار قدرتمند از more بوده و با اینکه دستور more هم در لینوکس وجود دارد ولی استفاده از آن به معنی عدم آشنایی شما با دستور بهتر less است. بدون جنگ با کسانی که از more استفاده می کنند، به راحتی از less استفاده کنید و فراموش نکنید که

```
less is more than more!
```

wc برای گرفتن آمار یک متن

دستور wc اسمش را از توالت نگرفته! wc مخفف word count یا شماره کلمات است. در صورت دادن یک فایل به عنوان آرگومان، این دستور به شما می‌گوید که این فایل حاوی چند کاراکتر، چند کلمه و چند خط است:

[1] سیستم عامل دیسک که اولین سیستم عامل مایکروسافت برای کامپیوترهای پی سی بود

```
jadi@wonderland:~$ wc ilovelist.txt
8  9 48 ilovelist.txt
```

مشاهده می‌کنیم که فایل **ilovelist.txt** حاوی ۸ خط، ۹ کلمه و ۴۸ کاراکتر است. سوییچ بسیار رایج این دستور **l-** بوده که از **line** گرفته شده و تنها می‌گوید فایل ورودی، چند خط دارد:

```
jadi@wonderland:~$ cat digits.txt
1
2
3
4
5
6
7
8
9
0
jadi@wonderland:~$ wc -l digits.txt
10 digits.txtd:$\sim$\$ wc -l digits.txt
10 digits.txt
```

grep برای جستجوی متن

با این دستور بسیار پرکاربرد، می‌توانیم در یک یا چند فایل به دنبال خطوطی بگردیم که حاوی عبارت خاصی هستند. برای اول به محتویات این فایل نگاه کنید:

```
jadi@wonderland:~$ cat sites
linux   linuxbook.ir
jadi    jadi.net
jadi    twitter.com/jadi
google  google.com
mail    gmail.com
github  github.com
```

```
github  github.io
```

حالا برای پیدا کردن تمام خطوطی که در آن کلمه **jadi** آمده می‌زنیم:

```
jadi@wonderland:~$ grep jadi sites
jadi    jadi.net
jadi    twitter.com/jadi
jadi@wonderland:~$
```

یا به دنبال تمام سایت‌هایی می گردیم که روی دامنه‌های دات کام هستند:

```
jadi@wonderland:~$ grep com sites
jadi    twitter.com/jadi
google  google.com
mail    gmail.com
github  github.com
```

لازم به تکرار است که مانند هر دستور دیگر لینوکس، استفاده از به جای نام فایل به معنای در تمام فایل‌های دایرکتوری جاری است پس با دستور زیر می‌توان در تمام دایرکتوری خانه جادی[1] ، به دنبال خطوطی گشت که **linux** در آن‌ها آمده:

```
jadi@wonderland:~$ grep -r linux ~
/home/jadi/ilovelist.txt:linux
/home/jadi/sites:linux  linuxbook.ir
```

sudo برای اجرای دستورات با دسترسی روت

این دستور به شما اجازه می‌دهد که با وارد کردن پسورد خودتان دستوری را با دسترسی روت یعنی بالاترین سطح دسترسی یک سیستم لینوکسی، اجرا کنید. در سیستم‌هایی مانند اوبونتو که به شکل پیش فرض کاربر روت اجازه داخل شدن به سیستم را ندارد، **sudo** روشی است برای کارهایی که نیازمند دسترسی بالاتر هستند. برای مثال با دستور زیر می‌توانید برنامه **jcal** را نصب کنید:

```
jadi@wonderland:~$ sudo apt-get install jcal
[sudo] password for jadi:
...
```

[1] فراموش که نکرده‌اید؟ هر کاربر با علامت ~ به دایرکتوری خانه‌اش اشاره می‌کند

```
...
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
libjalali0
The following NEW packages will be installed:
jcal libjalali0
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 32.4 kB of archives.
After this operation, 129 kB of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 http://us.archive.ubuntu.com/ubuntu/trusty
/universe libjalali0 amd64 0.4.1-2 [13.2 kB]
...
Setting up jcal (0.4.1-2) ...
Processing triggers for libc-bin (2.19-0ubuntu6) ...
```

کاملا واضح است که در همه سیستم‌ها همه کاربرها به این دستور دسترسی ندارند و زدن sudo ممکن است است به آن‌ها اخطار بدهد که آن‌ها اجازه ارتقاء دسترسی خود به روت را ندارند و این عمل مجرمانه آن‌ها به مدیر سیستم گزارش خواهد شد. اگر چنین اتفاقی برای شما افتاد نگران نباشید چون هیچ وقت هیچ کس به این گزارش‌ها توجهی نمی‌کند.

passwd

با پرسیدن پسورد فعلی، پسورد شما را تغییر می‌دهد.

```
jadi@wonderland:~$ passwd
Changing password for jadi.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

توجه کنید که در حین وارد کردن کلمه عبور قدیم و بعد کلمه عبور جدید، هیچ چیزی روی صفحه نمایش داده نمی‌شود تا کسی که ممکن است از بالای شانه شما در حال نگاه کردن به مانیتور باشد، متوجه پسورد شما و حتی طول آن هم نشود. در بعضی سیستم‌ها ممکن است حداقلی اجباری برای طول یا پیچیدگی پسورد در نظر گرفته شده باشد و اگر پسوردی کوتاهتر وارد کنید، پذیرفته نشود. همیشه پیام‌های سیستم را با دقت بخوانید تا متوجه موفقیت آمیز بودن یا خطای احتمالی کارهای خود بشوید.

دستورات عمومی

clear

یکی از ساده‌ترین دستورات: صفحه را پاک می‌کند تا محیط کار شما خلوت و تمیز شود.

date

در ساده‌ترین شکل، تاریخ فعلی سیستم را نشان می‌دهد.

```
jadi@wonderland:~$ date
Sun Jul  6 17:17:39 IRDT 2014
```

locate

این دستور جای یک فایل را به شما نشان می‌دهد. فرض کنید می دانید در سیستم فایلی به اسم `do.md` وجود دارد ولی نمی‌دانیم کجاست، دستور `locate` می توانید مکان آن را نشان دهد:

```
jadi@wonderland:~$ locate do.md
/home/jadi/w/fun/do.md
/var/lib/dpkg/info/sudo.md5sums
```

دقت کنید که یک فایل دیگر هم که حاوی این عبارت بوده پیدا شده. به خاطر پرهزینه بودن گشتن تمام فایل‌های سیستم، دستور `locate` به شکل زنده و در زمان واقعی روی دیسک جستجو نمی‌کند بلکه هر بیست و چهار ساعت یکبار با دستوری به نام `updatedb` فهرستی از همه فایل‌های سیستم می‌سازد و موقع اجرای دستور `locate` فقط به این بانک اطلاعاتی نگاه می‌کند. به عبارت دیگر در صورتی که فایلی به تازگی ایجاد شده باشد در خروجی‌های `locate` دیده نخواهد شد و اگر فایلی را از سیستم پاک کنید، تا بیست و چهار ساعت هنوز در خروجی دستور `locate` دیده می‌شود. برای به روز رسانی بانک اطلاعاتی می‌توانید به شکل دستی و با دسترسی روت دستور `updatedb` را اجرا کنید (مثلا با `sudo updatedb`.

cal

این برنامه می‌توانید یک تقویم زیبا را نشان دهد. در صورتی که بدون هیچ سوییچی از آن استفاده کنید، تقویم ماه جاری را نشان خواهد داد:

```
jadi@wonderland:~$ cal
July 2014
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
6  7  8  9  10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

ولی برای برنامه ریزی می‌توانید با دادن سوییچ ۳- از برنامه درخواست کرد که تقویم ماه‌های قبل و بعد را هم نشان بدهد:

```
jadi@wonderland:~$ cal -3
June 2014              July 2014              August 2014
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
1  2  3  4  5  6  7          1  2  3  4  5                  1  2
8  9  10 11 12 13 14   6  7  8  9  10 11 12   3  4  5  6  7  8  9
15 16 17 18 19 20 21   13 14 15 16 17 18 19   10 11 12 13 14 15 16
22 23 24 25 26 27 28   20 21 22 23 24 25 26   17 18 19 20 21 22 23
29 30                  27 28 29 30 31         24 25 26 27 28 29 30
31
```

و از این هم جالب‌تر، چیزی مانند `cal 1979` است که تقویم سال ۱۹۷۹ را نمایش می‌دهد.

> با نصب برنامه `jcal` ، تمام این قابلیت‌ها را در تقویم جلالی ایران خواهید داشت و مثلا من می‌توانم با زدن دستور `jcal 1356` کشف کنم که روز تولدم چند شنبه بوده است.

history

هزار دستوری که سابقا در سیستم اجرا شده‌اند را نشان می‌دهد.

عملیات مرتبط با کاربر

man

این دستور مهمترین دستور لینوکس است! مخفف manual بوده و با کمک آن می‌توانید راهنمای دستورات دیگر را مطالعه کنید. برای مثال وارد کردن دستور man ls راهنمای کامل دستور ls را به شما نشان می‌دهد. خواندن این راهنماها در اوایل کار کمی دشوار است اما با کمی تلاش می‌توانید آن‌ها را بفهمید، به محض درک شیوه خواندن صفحات man (که باکلاس‌ترها به آن man page می‌گویند) دیگر نیازی به هیچ راهنمایی نخواهید داشت - حداقل در سطح این کتاب.

اینجا سری اول از دستورات خط فرمان را تمام می‌کنیم. در درس بعدی می‌توانید دستورهای پیشرفته تر مانند دستورهای مرتبط با شبکه را ببینید و کشف کنید که چطور این دستورات ریز که هر کدام یک کار را به خوبی انجام می‌دهند، می‌توانند مثل لگو با هم ترکیب شوند تا ابزارهای جادویی خلق کنند.

۳.۳ استفاده از پروکسی در خط فرمان

منطقا باید منتقل بشه به خط فرمان ایران یکی از سانسور کننده‌ترین کشورهای جهان است و اینترنت در ایران دومین اینترنت سانسور شده دنیا. به همین خاطر اینترنت بدون پروکسی عملا در ایران کاربردی ندارد. البته پروکسی‌ها در دنیای بیرون از ایران هم کاربرد دارند. مثلا بسیاری از شرکت‌ها برای جدا کردن شبکه داخلی‌شان از اینترنت، از یک پروکسی استفاده می‌کنند و اگر شما بخواهید کاری با اینترنت انجام دهید، لازم است تنظیمات صحیح پروکسی را وارد کنید.

این روزها در گنوم و در کی.دی.ای امکان تنظیم مرکزی پروکسی فراهم است و بعضی برنامه‌ها هم خودشان این امکان را به شکل مستقل دارند. اما اگر مثل یک نینجای گنو/لینوکسی در خط فرمان باشید چه؟ در خط فرمان `Bash` ، این متغیرها می‌توانند وظیفه تنظیم پروکسی برای پروتکل‌های مختلف را بر عهده بگیرند:

```
HTTP_PROXY
HTTPS_PROXY
FTP_PROXY
GOPHER_PROXY
WAIS_PROXY
```

مثلا اگر من بخواهم با برنامه `apt-get` با استفاده از یک پروکسی چیزی نصب کنم باید بزنم:

```
$ sudo su -
# export HTTP_PROXY=http://proxy:port
# export HTTPS_PROXY=http://proxy:port
# apt-get install zim
```

اول کاربر را به کاربر ریشه تغییر می دهم، بعد متغیرهای مربوط به پروکسی را مقداردهی و اکسپورت می کنم (اگر متوجه نشده‌اید که به جای پروکسی و پورت باید مقدار مورد استفاده خودتان را بگذارید شاید این راهنما برای شما نوشته نشده باشد) و بعد برنامه را مطابق معمول نصب می‌کند - اما حالا برنامه `apt-get` در هنگام استفاده از پروتکل های `http` و `https` با نگاه به متغیرهای مربوط به آن‌ها، از پروکسی مرتبط استفاده خواهد کرد[۱].

[۱] برای ترمینال از پروکسی همراه معتبرسازی استفاده کنیم `cyletech.blogspot.de`

۴.۳ چگونه یک دامین و یک هاست را به یکدیگر متصل کنیم

برای اضافه کردن یک ساب دامین به یک دامین یا به عبارت بهتر برای اضافه کردن یک دامین به یک هاست، نیازمند قدم های مختلفی هستیم:

تنظیم دی ان اس ها روی آن دامین یا ساب دامین = گفتن اینکه در صورت درخواست فلان دامنه (مثلا `jadi3.undo.it`) اینترنت باید درخواست کننده را به هاست ما هدایت کند تنظیم وب سرور برای پاسخ مناسب دادن به آن دامین توجه کنید که این جریان هیچ ربطی به مفهوم ساب دامین ندارد. ساب دامین هم درست مثل دامین عمل می کند: یک اسم است که توسط یک سرویس به نام DNS که وظیفه تبدیل آدرس های انسان-فهم اینترنتی به عددهای آی پی کامپیوتر-فهم را بر عهده گرفته. من شخصا از سرویس های دی ان اس رایگان روی اینترنت از جمله `freedns.afraid.org` برای کارهای عمومی ام استفاده می کنم. پس اولین قدم برای من رفتن به آنجا، رجیستر کردن دی ان اس مورد نظر و ارسال آن به آی پی سرویس دهنده وب است.

حالا هر کس بزند `jadi3.undo.it` از آنجا به وب سرور من هدایت خواهد شد. قدم بعدی این است که به وب سرورم بگویم هر کس به این آدرس آمد کدام دایرکتوری به عنوان محتویات آن آدرس وب به او نمایش داده شود. من قبلا در تنظیمات آپاچی (که مثل همه تنظیمات دیگر در `/etc` قرار دارد) داشتم:

```
ServerAdmin jadijadi@gmail.com
DocumentRoot /home/jadi/public_html/
ServerName www.freekeyboard.net
ErrorLog /home/jadi/www-error_log
```

```
CustomLog /home/jadi/www-access_log combined
```

و حالا فقط کافی است این را هم اضافه کنم که:

```
ServerAdmin jadijadi@gmail.com
DocumentRoot /home/jadi/public_html/
ServerName jadi3.undo.it
ErrorLog /home/jadi/www-error_log
CustomLog /home/jadi/www-access_log.jadi common
```

تا به وب سرور گفته باشم که اگر کسی اومد و با آدرس `jadi3.undo.it` کار داشت بهش محتویات فلان دایرکتوری رو نشون بده. می بنین که من اینجا همون دایرکتوری قبلی رو معرفی کردم و در نتیجه هر کس آدرس `http://jadi3.undo.it` رو بزنه به همون جایی می رسه که بقیه دامین های من می رسن. مشخصه که هر تغییری در فایل تنظیمات نیازمند ری استارت سرویس است. سرویس آپاچی رو ری استارت می کنم:

```
# /etc/init.d/apache2 restart
```

نکات مهم در مورد این مقاله - دامین `undo.it` متعلق به من نیست. یک دامین آزاد است برای هر کسی که دوست دارد روی آن ساب دامین تعریف کند - تلفظ صحیح دامین ظاهرا دومین است - این مقاله نمی تواند راهنمای اینکار برای کسی باشد که درک نمی کند چکار می کند. دو قدم تنظیم دی ان اس و تنظیم وب سرور در هر محیط ممکن است به شکل متفاوتی انجام شود - این روش برای مبارزه با سانسور توصیه نمی شود (: اول سانسورچی اینقدر کم خرد است که تمام اینترنت را سانسور کرده پس همه از فیلتر رد می شوند و دوم اینکه من قرار نیست با چهار نفر مشتری سایت هر روز در حال فرار باشم. طبق تجربه من سانسور حداکثر ده درصد خواننده ها را کم کرده و در عوض ماندن روی همین دامین هر روز خواننده های جدید را اضافه می کند. - سرویس های دی ان اس درکل اینترنت باید پخش بشن. یعنی در لحظه تعریف سرور دی ان اس من می دونه که صاحب فلان دامین است و باید اونو به فلان آدرس بفرسته ولی طول می کشه اینو همه بفهمن. گفته می شه که لازمه بیست و چهار ساعت فرصت بدین تا همه دنیا بفهمن که این آدرس باید بره فلان جا - اسم های دامنه رایگان (مثلا همین آندو دات ایت) بامزه هستن ولی خیلی خوب نیستن. مثلا اگر چیزی روی این آدرس ها درست کنین ممکنه از نظر خیلی از سایت ها ذاتا اسپم باشه (:

فصل ۴

جامعه لینوکس

۱.۴ کمک گرفتن و ادامه راه

وجود هزاران نفر مشتاق کمک به شما، یکی از مزایای اصلی نرم‌افزارهای آزاد نسبت به نرم‌افزارهای تجاری و انحصاری است. البته در نرم‌افزارهای تجاری - مثلا ویندوز - هم شما معمولا به پشتیبانی دسترسی دارید اما این پشتیبانی در اکثر موارد ناکارا است. به دو دلیل:

کسی که سعی می‌کند از طریق تلفن پشتیبانی شما را بر عهده بگیرد در اکثر موارد از شما سواد کمتری دارد، انگلیسی بدتری حرف می‌زند و تنها سعی می‌کند از طریق پیگیری یکسری الگوریتم و نمودار (شبیه چیزی که موقع پیگیری `Troubleshooting` خود ویندوز می‌بینید) مشکلات شما را حل کند. اول از شما می‌پرسد که آیا کابل پرینتر وصل است. بعد می‌پرسد که آیا درایور نصب شده و بعد از شما می‌خواهد چک کنید که کاغذ در پرینتر گیر نکرده باشد. بعد ویندوز را ری‌استارت می‌کند و در صورت حل نشدن مشکل از شما خواهد خواست که کامپیوتر را به پیش یک تعمیرکار ببرید.

سیستم‌های بسته‌ای مانند ویندوز، چندان قابل «عیب‌یابی» نیستند. شما با یک صفحه آبی روبرو می شوید و بعد باید کامپیوتر را خاموش و روشن کنید. یک پیام می‌آید که به شما می‌گوید «مشکل جدی. لطفا به مایکروسافت اطلاع دهید» (و شما هم `Dont Send Error` را فشار می‌دهید) و ...

اما لینوکس بر خلاف ویندوز برای عیب یابی طراحی شده. در صورت بروز هر مشکل احتمالا چندین و چند `log` مختلف به نمایش درمی‌آید و اشکال هم قابل تکرار کردن خواهد بود و شما می‌توانید قدم به قدم مشکل را حل کنید. اینکه کجا متوقف شوید فقط بستگی به سواد، علاقه شما به یادگیری و حوصله‌ای که به کار می‌برید دارد. حالا برویم سراغ اینکه از چه جاهایی می‌توانید کمک بگیرید:

جستجو در اینترنت

این تقریبا مهمترین جا برای یافتن جواب سوالات است. می‌خواهید بدانید چطور می‌توانید آی.پی. استاتیک به کارت شبکه بدهید؟ `how to configure static ip` را گوگل کنید. می‌خواهید کشف کنید که چطور باید اوبونتوی ۰۴.۹ خود را به ۱۰.۹ آپگرید کنید؟ دنبال `upgrade 9.04 to 9.10` بگردید. از صفحه لاگین خود خسته شده اید و می‌خواهید تصویرش را تغییر دهید؟ « `change login screen linux` » (: ساده نیست؟ کاملا ساده و سر راست.

در هنگام جستجو از کلمات کلیدی دقیق استفاده کنید. اگر دقیقا سوالتان مربوط به عملکرد یک توزیع خاص (مثلا سابایون یا مندرویا) است، آن را به عبارت مورد جستجو اضافه کنید و در غیر اینصورت، به شکل عمومی از `linux` استفاده کنید. از انگلیسی نوشتن و خواندن نترسید و قدم به قدم جلو بروید. سعی کنید بفهمید راه حل ارائه شده از چه طریقی کار می‌کند و حداقل به چند نتیجه جستجو توجه کنید و یکضرب امیدوار یا ناامید نشوید.

رفتن به انجمن‌ها

فروم‌ها قدم دوم هستند. در فروم‌ها ده‌ها، صدها یا هزاران نفر هستند که به دلایل مختلف (از پیشبرد جامعه و اعتقاد به ثواب در آن دنیا تا اظهار فضل و کسب شهرت و حتی احساس لذت از گیک بودن) حاضرند به شما کمک کنند. معمولا از نظر فرهنگی بهتر است در فرومی که سوال می‌کنید، یک شناسه تعریف کنید و بسیار دقیق و مودب سوال خود را مطرح کنید. البته قبل از پرسیدن هر چیزی باید خوب جستجو کنید تا ببینید آیا همین سوال یا سوالی خیلی نزدیک به آن قبلا پرسیده شده یا نه.

کسانی که به شما کمک می‌کنند، داوطلب هستند پس سعی کنید شما هم حداقل با پرسیدن سوال دقیق و «هوشمندانه»، به آن‌ها کمک کنید. در عین حال مطمئنا در هر سطحی که باشید، کسانی هستند که نیازمند کمک شما هستند. بسیار خوب است که خودتان هم به نوبه خود به آن‌ها کمک کنید.

در ایران فروم‌های خوبی وجود دارند. حرفه‌ای‌ترین و قدیمی ترین انجمن، انجمن تکنوتاکس است. انجمن اوبونتوکاران ایران هم این روزها یکی از فعالترین و پرکابرترین فروم‌ها است و بقیه توزیع‌ها هم فروم‌های خود را دارند که معمولا می‌توانید به سادگی با یک جستجو آن‌ها را پیدا کنید.

در سطح جهان هم تعداد خیلی زیاد فروم وجود دارد که یک جستجوی ساده در گوگل، فعالیترین آن‌ها را به شما نشان خواهد داد.

IRC

یک خاطره برای همه گیک‌های قدیمی و هنوز هم اصلی‌ترین کانال ارتباطی حرفه‌ای با یکدیگر. آی.آر.سی. یا اینترنت ریلی چت، یک جور اتاق گفتگو است که در آن می‌توانید با کسانی که در همان لحظه در همان اتاق هستند صحبت کنید. یک نفر همزمان می‌تواند درون اتاق‌های مختلفی از سرورهای مختلف باشد و معمولا هم سریعترین و حرفه‌ای ترین جواب در این اتاق‌ها گرفته می‌شود. آی.آر.سی. فرهنگ خاص خودش را دارد. در آنجا نباید قبل از اجازه گرفتن از کسی، به او پیام خصوصی بدهید. نباید چیزی بیشتر از دو سه خط را در اتاقی عمومی پیست کنید و به هیچ وجه نباید از فونت های عجیب و غریب و زبان‌های محلی و ... استفاده کنید. در آی.آر.سی. مثل هر جای دیگر مودب باشید و بدانید که اگر کسی به شما کمک می‌کند یک داوطلب بدون چشمداشت است و شما هم باید در حد توان به دیگران کمک کنید.

برای اتصال به یک سرور آی.آر.سی. باید از برنامه‌های ویژه اینکار استفاده کنید.

در ایران این کانال‌های مشهور آی.آر.سی. فعال هستند:

- تکنوتاکس

فصل ۵

زندگی حرفه‌ای

۱.۵ لینوکس به عنوان شغل

حرفه‌ای را من دقیقا به از «حرفه» می‌گیرم و منظورم «وارد» نیست. منظورم از «حرفه ای لینوکس» دقیقا کسی است که از لینوکس درآمد دارد و مطمئن هستم که خیلی‌ها دوست دارند به اینجا برسند.

یک نفر لینوکس کار، می‌تواند مشاغل مختلفی داشته باشد. مثلا مدیر سیستم، مشاور امنیت، پشتیبان فنی و غیره. در عین حال این احتمال هم هست که لینوکس به شما کمک کند در شاخه دیگری شغل بهتری پیدا کنید، مثلا یک مهندس مخابرات در حوزه موبایل تقریبا همیشه با سیستم‌های مبتنی بر لینوکس سر و کار دارد و در نتیجه مهندس مخابراتی که در گنو/لینوکس مهارت داشته باشد، بازار کار بسیار بهتری از مهندس حتی بهتری که به گنو/لینوکس وارد نیست دارد.

این سوال یک سوال همیشگی است: «اگر لینوکس آزاد/رایگان است، پس متخصصانش از کجا پول در می آورند؟». راستش من هیچ وقت این سوال را نفهمیده ام. منظور چیست؟ مگر حقوق یک برنامه نویس ویندوز یا مدیر سیستم ویندوز به خاطر قیمت ویندوز پرداخت می‌شود؟ لینوکس رایگان است. این یعنی افراد بدون هزینه حق دارند کرنل لینوکس و خیلی از توزیع های مهم گنو/لینوکس را از اینترنت دانلود کنند. این چه ارتباطی دارد به اینکه یک نفر حقوق بگیرد و فلان سرویس را روی فلان توزیع بالا بیاورد یا مدیریت کند؟ این مساله حتی در مورد کسانی که نرم افزار آزاد می‌نویسند هم چندان باربط نیست.

تصور دانشجوها و کسانی که هنوز به محیط های کار واقعی نرفته‌اند از شغل، این است که برنامه ای نوشته می‌شود و محصول نهایی به فروش می‌رود و آدم ها پول در می‌آورند. در حالی که درصد خیلی کمی از برنامه‌نویس‌های دنیا هستند که برنامه ای بنویسند که در بازار فروش رود و آن ها از پول فروش سهمی بردارند. اکثر برنامه نویس‌های دنیا، «استخدام» می‌شوند تا برنامه بنویسند. آن‌ها حقوق ثابتی دارند و در اکثر موارد هم نتیجه نهایی، یک برنامه تک منظوره است که در بازار به مصرف کننده نهایی فروخته نمی‌شود. یک سر به بازار نرم افزارهای کامپیوتری بزنید و ببینید کلا چند برنامه قابل خریدن در دنیا وجود دارد. بعد آن را مقایسه کنید با حجم عظیمی کامپیوتر که در جهان کارهای روزمره را انجام می‌دهند: وب سایت‌های خرید آنلاین، برنامه های کنترل راه آهن، برنامه ایمیل آنلاین، فیسبوک، کنترل کننده چراغ های راهنمایی، سایت های دوست یابی، قهوه جوش های قابل برنامه‌ریزی، خبرگزاری ها، سیستم اتوماسیون اداری، . می بینید؟ اکثر برنامه نویسان جهان از فروش مستقیم نتیجه کارشان در بازار پول در نمی‌آورد بلکه حقوق می گیرند، چه برسد به مدیران سیستم و متخصصان سیستم عامل.

اما یک متخصص لینوکس از چه کارهایی ممکن است پول در بیاورد؟

مدیریت سیستم

این بدون شک اصلی ترین شغل یک متخصص گنو/لینوکس است. کسی که مواظب سرورها است. کنار آن‌ها راه می‌رود، آن‌ها را آپدیت می‌کند. به چراغ‌های خطر آن‌ها توجه می‌کند. سیستم عامل‌ها را تنظیم می‌کند.

بک آپ می‌گیرد و در صورت نیاز بک آپ‌ها را بازیابی می‌کند و اینجور کارها.

یک مدیر سیستم معمولا مسوول درست کار کردن سیستم‌ها است و پاسخ دادن به نیازهای روزمره کامپیوترها. اما حالت دیگری هم وجود دارد. گاهی مدیر سیستم به کسی اطلاق می‌شود که شغلش نصب و راه اندازی سیستم است.ممکن است از شما بخواهند که کامپیوتری که جدیدا خریداری شده را در رک نصب کنید، رویش لینوکس بریزید، تنظیمات شبکه را انجام دهید و بعد آن را به یک کلاستر دیتابیس متصل کنید.

شاید هم از شما بخواهند روی کامپیوتری که روی رک نصب شده `raid` کانفیگ کنید، رویش لینوکس بریزید، آن را به شبکه نصب کنید و بعد یک وب سرور رویش راه بندازید که از SSL هم پشتیبانی کند. بعد از اینکه همه این کارها را کردید، یا خودتان مسوول نگهداری سیستم می‌شوید یا یک مدیر سیستم دیگر.

پشتیبانی سیستم

شرکت‌هایی هم هستند که از پشتیانی سیستم پول درمی‌آورند. کافی است یک شرکت باز کنید و بهترین متخصصان را استخدام کنید. حالا به شرکت‌های بزرگ قول بدهید که در قبال ماهی مثلا ۵ هزار دلار، تمام مشکلات مرتبط با لینوکسی که مهندسان خود شرکت نتوانند حل کنند را حل می‌کنید.

فروش سیستم عامل

درست است که گنو/لینوکس آزاد است اما هیچ کجای مفهوم نرم افزار آزاد گفته نشده که کسی حق فروش آن را ندارد. مشهورترین نمونه شرکت ردهت است که لینوکس ردهت را تولید می‌کند. شما حق دارید سورس این توزیع را از اینترنت بگیرید و شخصا کامپایل کنید اما هنوز هم که هنوز است اکثر شرکت‌های بزرگ دنیا ترجیح می‌دهند با وجود حضور ده‌ها توزیع قوی و رایگان سرور، هزینه نسبتا بالایی پرداخت کنند تا با داشتن ردهت، از پشتیبانی فنی آن نیز بهره ببرند. جدیدا نیز شرکت‌هایی مثل زوزه و بعد اوبونتو، همین مسیر را در پیش گرفته‌اند.

برنامه‌نویسی روی لینوکس

برنامه نویسی روی لینوکس تفاوت چندانی با برنامه‌نویسی روی پلتفرم‌های دیگر ندارد اما تعداد برنامه‌نویسانی که اینکار را بلدند کمتر است. این یعنی بازار کار بهتر حتی با وجود کوچکتر بودن تقاضا. اگر کسی برنامه‌نویسی بر روی لینوکس را به خوبی بلد باشد، احتمال زیادی دارد که شغل خوبی پیدا کند - البته با جستجویی بیشتر :)

آموزش

اگر در موسسه‌ای لینوکس درس بدهید، مشغول پول درآوردن از لینوکس هستید. این روزها پول واقعی در تدریس است و کسانی که می‌خواهند - برای درآمد بهتر - لینوکس یاد بگیرند، معمولا حاضر هستند پول بیشتری هم هزینه کنند. یک کمپ یک هفته‌ای لینوکس برای آمادگی امتحان ال پی آی، تقریبا چهارصد هزار تومان هزینه دارد. فرض کنید فقط ده نفر در کلاس باشند؛ این یعنی چهار میلیون تومان در یک هفته. مطمئنا هر هفته کلاس ندارید و پول کلاس هم فقط به مدرس نمی‌رسد اما ...

نویسندگی

برای این کتاب پولی نداده‌اید ولی اگر روی کاغذ چاپ می‌شد، باید برای خواندنش پول خرج می‌کردید و تقریبا بیست درصد هزینه پشت جلد به من می‌رسید. فرض کنید ۱۰۰۰ نسخه از کتابی که نوشته اید به قیمت ۵۰۰۰ تومان به فروش برود و شما از چاپ اول یک میلیون تومان درآمد خواهید داشت. زیاد نیست ولی بانمک است. اگر با کتاب و مشارکت در سایت‌ها هم مشهور شده باشید، مجلات و روزنامه‌ها با علاقه مقالات شما در مورد لینوکس را چاپ خواهند کرد که بازهم در ایران درآمد کمی است اما برای یک دانشجو جذاب (:

لینوکس به عنوان ارزش افزوده در مشاغل دیگر

اما این همه داستان نیست... شغل‌هایی هم هستند که مستقیما لینوکس نیستند اما در آن‌ها از لینوکس به عنوان ارزش افزوده نام برده می‌شود.

در بخش قبل، شغل‌های لینوکسی، نگاهی به شغل‌هایی انداختیم که مستقیما از لینوکس بلد بودن منتج شده‌اند. اما به نظر شخصی من، یاد گرفتن لینوکس در بیشتر موارد منجر به شغل‌هایی می شود که در ظاهر ارتباط مستقیم با لینوکس ندارند اما اگر کسی لینوکس بلد باشد هم شانس بسیار بیشتری برای استخدام شدن در آن‌ها دارد و هم احتمالا حقوقی بالاتر.

مثلا در دنیای موبایل، تقریبا همه سرویس‌ها (از جمله اس ام اس، جی پی آر اس، خدمات خط به خط و غیره) وابسته به سرورهای لینوکسی هستند و اگر یک نفر مهندس مخابرات بخواهد در این حوزه کار کند، بدون بلد بودن گنو/لینوکس تقریبا شانسی نخواهد داشت. مساله مشابهی را می‌شود در مورد شغل طراحی وب مثال زد. یک طراح وب مستقیما با لینوکس کاری ندارد اما طراح وبی که با لینوکس آشنا باشد، به احتمال زیاد شانس بیشتری برای استخدام شدن نسبت به طراحی دارد که با این سیستم عامل آشنا نیست. همین مساله به راحتی ممکن است در مورد برنامه‌نویس‌هایی هم پیش بیاد که برنامه‌های خاص لینوکس نمی‌نویسند اما محیط توسعه آن‌ها لینوکسی است.

علاوه بر این، مشارکت در دنیای لینوکس می‌تواند به راحتی برای شما شهرت و اعتبار هم بیاورد. اگر شما فقط ده خط برنامه داشته باشید که به جایی از کرنل لینوکس اضافه شده باشد، بدون شک رزومه شما - برای

یک شغل مرتبط - بهتر از روزمه هر کسی است که در ده دوره برنامه نویسی C شرکت کرده باشد و بیست و پنج مدرک برنامه نویسی گرفته باشد.

حتی اگر شما برنامه‌نویس نباشید، با نوشتن یک کتاب، با مشارکت در انجمن‌ها، با نوشتن بررسی‌های لینوکسی و با همکاری با مجله‌ها به عنوان یک متخصص معروف می‌شوید و این شانس شما را برای گرفتن کارهای بهتر (حتی نامرتبط) افزایش خواهد داد.

۲.۵ آیا به دانشگاه بروم

این سوال رو زیاد می پرسن. راستش رو بخواین من هم زمان کنکورم بود یکی از Day Dream هام این بود که بیخیال کنکور و دانشگاه بشم و برم سراغ شرکت باز کردن (دقیقا نمی دونستم چی! فقط اسم شرکت رو می گفتم به عنوان یک شغل) یا مدرک های لینوکس گرفتن یا هر راه فرار دیگه ای برای خلاصی از کنکور.

صادق باشیم: در بسیاری از مواقع، اینها راه فرار هستن نه نقشه واقعی. متاسفانه رفتن به دانشگاه فعلا معقول ترین مسیر پیشرفته؛ بخصوص توی ایران. دلایلش هم خیلی ساده است. به چند تاش اشاره می کنم.

مهمترین دلیل اینه که دوران دانشجویی دوران خیلی خاصیه. نه خانواده از شما انتظار خاصی داره نه جامعه. می تونین هر جینگولک بازی ای که می خواین رو در بیارین (از بلند کردن مو گرفته تا انقلاب کردن تا نقد تا فاز هنری گرفتن تا سفر شمال تا ...) و مدعی بشین که «دانشجو» هستین (: دوران تحت کنترل کامل خانواده بودن گذشته و دوران مسوولیت مالی داشتن هم هنوز نرسیده. این دوران خاص رو از دست ندین

دومین دلیل اینه که دانشگاه اولین محیط مختلط است که بالاخره حکومت ما -کمابیش- قبول می کنه دو نفر آدم حق دارن کنار هم باشن و با هم حرف بزنن. این رو هم الکی از خودتون نگیرین. زمین تا آسمون با پارتی مختلط و اینها متفاوت است. یک تعامل واقعی با جنس مقابل.

بعدش اینکه در صورت علاقمندی به مهاجرت مدرک دانشگاه بسیار بسیار کمک کننده است. بدون مدارک دانشگاهی تقریبا مهاجرت غیرممکن می شه. البته من طرفدار موندن تو ایران هستم ولی به هرحال اکثریت دنبال مهاجرت هستن و دانشگاه یکی از راحت ترین روش هاش.

اینم بگم که حتی اگر بخواین در خیلی از کشورها کار کنین، اون کشورها معمولا برای دادن ویزای کار ازتون ترجمه مدرک مرتبط دانشگاهی می خوان. حتی اگر شرکت خودتون بخواد شما رو بفرسته افغانستان کار کنین هم سفارت ازتون مدرک مرتبط با کار می خواد. چه برسه به قطر و غیره (:

می‌بینین؟ شاید از نظر عقلی بشه گفت که در سطح علمی مدارک معتبر گرفتن بهتر از دانشگاه رفتنه ولی در دنیای واقعی دور و بر ما، دانشگاه اهمیت خیلی خاصی داره که هر استدلالی که «دانشگاه نرم فلان کار رو بکنم» رو تبدیل می کنه به یک بهانه برای فرار از سختی‌های رفتن به دانشگاه. درسته که این روزها ورود به یکسری از دانشگاه ها خیلی سخته و سال های خوبی از زندگی رو حروم می کنه ولی اگر تصمیم می گیریم نریم دانشگاه لازم نیست دنبال بهانه هایی مثل «دانشگاه مفید نیست» قایم بشیم (: خب روراست بگیم حوصله نداریم دو سال جون بکنیم و رقابت کنیم.. در ضمن چیزهایی مثل دانشگاه آزاد رو هم خدا آفریده دیگه ورود بهشون راحت تره اسمشون هم دانشگاهه و یکسری از موارد بالا رو هم پاسخگو.

مغلطه: بیل گیتس و جابز و زوکربرگ دانشگاه رو تموم نکردن

نمی خوان بزنم تو ذوقتون ولی شما بیل گیتس و زوکربرگ و جابز نیستین (: اونها در شونزده هفده سالگی مثل یک گیک دنبال درس بودن نه دنبال دلیل برای نرفتن به دانشگاه. همه وارد دانشگاه شدن و بعد وسطش

اونو ول کردن. در ضمن میلیون ها نفر دیگه هستن که دانشگاه نرفتن یا دانشگاه رو ول کردن و من و شما اسمشون رو هم نشنیدیم. در اصل این سه تا و بقیه هم‌پالکی‌هاشون هم چون این سه تا بودن دانشگاه رو ول کردن نه اینکه چون دانشگاه رو ول کردن این سه تا شدن (:

بحث این نیست که اگر کسی نره دانشگاه موفق نمی شه یا دانشگاه ملاک موفقیته یا هر چی. بحث اینه که در یک بررسی معقول دانشگاه رفتن هم توی دید، هم توی زندگی شخصی و هم توی زندگی حرفه ای مفیده. اگر کسی نمی خواد بره حق داره نره و بره دنبال راه خودش. خیلی هم عالیه و قابل افتخار اما اگر حس کردین دارین پشت این استدلال قایم می شین تا از یک چیز سخت به اسم کنکور فرار کنین، یک جای کار می لنگه (:

معیار شخصی

من یکسری معیار شخصی دارم. یکیش اینه که «کجا و چه زمانی چی می گم». اگر دارم سال آخر که باید درس خوندن جدی برای کنکور رو شروع کنم توضیح می دم «دانشگاه خیلی هم خوب نیست» کمی مشکوکه جریان درست همونطور که اگر درست همون موقع که توی آب چاه خونه ام قرآن افتاده فتوا می دم که «خوردن آب از چاهی که توش قرآن افتاده مشکلی نداره» جریان مشکوکه، با خودتون فکر کنین که چرا دقیقا الان که جریان سخت شده مغزم هی داره سعی می کنه متقاعدم که کنه که «اصلا دانشگاه به درد نمی خوره». اگر واقعا معتقدم آدم موفقی هستم چرا الان که این استدلالها هست نرم دانشگاه و وقتی دقیقا برنامه داشتم ازش نیام بیرون؟ به عبارت دیگه هر وقت حرفی زدم باید ببینم «در کجای جهان ایستاده ام» که اینو می گم (: اگر درست پشت در امتحان کنکور دارم توضیح می دم که دانشگاه اصلا خوب نیست کاملا فرق می کنه با حالتی که توی دانشگاه در حال این استدلال باشم[1].

[1] برای بحث بیشتر به این مجموعه کامنت مراجعه کنین jadi.net

۳.۵ رزومه

هرچقدر هم که با یک نفر نزدیک باشید یا هر چقدر که خودتان را برای کاری مناسب بدانید، اولین ریپلای به ایمیل درخواست کار این است «رزومه»! نوشتن رزومه در ابتدا کاری بسیار سخت است. اگر هیچ رزومه ای ندارید بهتر است درست بعد از خواندن این متن، یک رزومه برای خودتان دست و پا کنید چون در دنیای حرفه‌ای کار، رزومه شما نه فقط معرف که اولین قدم در گرفتن هر نوع کاری است. رزومه بهتر است در یک فرمت مرسوم و طبیعی مثل برنامه‌های صفحه پرداز آفیس، فایل‌های مارک داون، تخ و ... نوشته شود و در هنگام ارائه به هر کس به پی.دی.اف. تبدیل شود. نکات زیر اصلی‌ترین چیزهایی است که در دنیای رزومه نویسی به نظر من می‌رسد. البته مشخص است که بعد از نکات بدیهی‌ای مانند ذکر روش تماس و آوردن تحصیلات و سابقه کاری از جدید به قدیم.

اطلاعات بی‌ربط ننویسید

این موضوع را بالاتر از همه آوردم چون معمولا در بالای رزومه اطلاعات بی‌ربط زیاد مشاهده می‌شود. در یک رزومه حرفه‌ای در سطح جهان حتی اشاره به سن و جنس هم بی‌ربط به حساب می‌آید چون شرکت‌ها حق ندارند بر اساس سن افراد، در مورد استخدام آن‌ها تصمیم بگیرند چه برسد به وضعیت ازدواج و سربازی و دین و قد و اسم پدر و شماره ملی و غیره. بخش اول رزومه باید معرف هویت شما به شکل کلی و روش‌های تماس با شما باشد. انتخاب اینکه به این بخش عکس اضافه کنید یا خیر با شماست.

خوب است درست زیر بخش اطلاعات فردی، توضیح مختصری در مورد خودتان و اینکه چرا به این شغل خاص علاقمند هستید ذکر کنید.

آپدیت کنید

همانطور که از نکته قبلی استنتاج می‌شود، رزومه شما بهتر است برای هر درخواست کار آپدیت شود. اگر من برای شغل «مدیر بخش مونیتورینگ» یک شرکت اپلای می‌کنم، بهتر است تجربیاتی مثل کار کردن در ساپورت یا تخصص در سیستم نگیوس پر رنگ تر باشد و مهارت‌های برنامه‌نویسی‌ام کمرنگ‌تر شود.

در مقابل اگر علاقمند شدم برای رفتن به شرکت فلان، نقش «تحلیل‌گر ارشد» را بپذیرم، سریعا تجربه‌ام در کار در بخش پشتیبانی لایه دو را کم‌رنگ می‌کنم و تاکیدم را روی مشارکت در توسعه سیستم نرم‌افزاری فلان بانک می‌گذارم.

این سرفصل می‌تواند یک معنای دیگر هم داشته باشد. حتی در دورانی که دنبال شغل نیستید هم رزومه‌تان را آپدیت کنید. اگر پوزیشن شما تغییر کرد، مدرک جدیدی گرفتید، مهارتی کسب کردید که نیاز به ذکر دارد و غیره همان موقع در فایل اصلی رزومه این را اضافه کنید تا در آینده راحت باشید.

فقط لینک ندهید

هر چقدر هم که در جاهای مشهوری کار کرده باشید یا سایت‌های جالبی طراحی کرده باشید، کمتر پیش می‌آید که مسوول بررسی رزومه آدرس سایت‌هایی که کار کرده‌اید را از روی کاغذ در کامپیوتر تایپ کند یا حتی سعی کند بفهمد منظورم من از ذکر اینکه مدرک مهندسی مخابرات‌ام را از دانشگاه KNTU گرفته‌ام، همان دانشگاه خواجه نصیرالدین طوسی است. بسیار خوب است که جلوی اسم شرکت‌ها، پروژه‌ها، دانشگاه‌ها، سایت‌ها و موارد مشابه چند کلمه توضیح اضافه کنید. مثلا بگویید «تحلیل‌گر در شرکت ایمپک (لهستانی و ارائه دهنده نرم‌افزارهای انتقال پول در شبکه‌های موبایل)».

خلاصه و خوانا بنویسید

حتما خوب است بخش‌هایی از رزومه با جملات کامل نوشته شده باشند اما در حالت‌های مرسوم اکثریت رزومه به شکل فهرست‌های عددی یا بولت‌لیست و جدول است که بتوان آن را به راحتی مرور کرد. در عین حال لازم نیست تک تک چیزهایی که بلد هستید را ذکر کنید. مثلا من دیگر در رزومه‌ام نمی‌گویم HTML بلد هستم چون هم از حوزه کاری من خارج است و هم باعث می‌شود چیزهای دیگر به اندازه کافی به چشم نیایند. در سرتیتر بعدی، مفصل‌تر به این موضوع می‌پردازم.

سطح مهارت‌هات را صادقانه بیان کنید

هر کس مقدار عظیمی چیز بلد است. اگر مدرک نرم افزار دارید بدون شک حداقل زمانی در مورد برنامه نویسی، اسمبلی، معماری کامپیوتر، دیتابیس، ساختار داده و ... چیزهایی بلد بوده‌اید. در یک رزومه حرفه‌ای لازم نیست به تک تک این موارد اشاره کنید. من بارها رزومه‌هایی دریافت می‌کنم که نویسنده‌اش مدعی است «سی، جاوا، سی پلاس پلاس، سی شارپ، جاوااسکریپت، نود، پی اچ پی، پایتون، و این روزها گو، روبی، بیسیک و پاسکال» را می‌داند. شخصا موقعی که صحبت از استخدام یک جاوا یا پی اچ پی کار حرفه‌ای است، هیچ توجهی به چنین رزومه‌ای نمی‌کنم. فراموش نکنید که «برنامه نویسی چیزی بیشتر از دانستن دستور یک زبان است» و برای هر درخواست رزومه‌تان را آپدیت کنید. همچنین لازم است جلوی هر مهارت (از زبان گفتاری تا زبان برنامه نویسی) میزان مهارت‌تان را به شیوه‌ای ذکر کنید.

> البته بعضی‌ها معتقد هستند اشاره به همه جزییات رزومه بهتری تحویل می‌دهد. بخصوص در مواردی که یک ماشین در مرحله اول رزومه‌ها را بررسی می‌کند و مثلا ممکن است هر کسی که عبارت `Office` در رزومه‌اش نیست، به کلی کنار گذاشته شود. من با این سبک موافق نیستم و انتخاب صد در صد با شماست.

نقش خودتان در بخش‌های مختلف را شرح دهید

فرض کن من در رزومه نوشته باشم «معمار سیستم، شرکت آتی‌تل، ۱۳۹۴ تا امروز». آیا ایده‌ای دارید که من در این شرکت چکار می‌کنم؟ در هر مورد لازم است حتی در یک پارگراف کوتاه در زیر هر سابقه شغلی، توضیح بدهید که دقیقا چه نقشی داشتید، روی کدام پروژه‌ها کار کرده‌اید و خروجی آن‌ها چه بوده. توضیح صادقانه، ساده و بدون اغراق این مساله چیزی است که تقریبا هر کارفرمایی را جذب خواهد کرد و باعث خواهد شد هر کسی که رزومه شما را ببینند، تا این بخش‌ها را کامل نخوانده کاغذ را زمین نگذارد.

ساده و بدون غلط بنویسید

به روز نگه داشتن و مدیریت یک رزومه فارسی و یک رزومه انگلیسی کار راحتی نیست. به همین دلیل من شخصا فقط یک رزومه ساده انگلیسی دارم. البته در مواردی ممکن است رزومه انگلیسی «کلاس گذاشتن» حساب شود یا بخش مدیریت منابع انسانی شرکت از شما رزومه فارسی بخواهد. این انتخاب شماست ولی هر کدام را که انتخاب می کنید رزومه **باید** بدون اشتباه و سلیس باشد. اگر انگلیسی می‌نویسید لازم نیست پیچیده بنویسید. از کلمات و جملات ساده استفاده کنید و در مرور زمان، جملات را اصلاح کنید.

در رزومه هکر نباشید

بعله! حتی اگر بزرگترین هکر هستید حداقل در رزومه هکر نباشید. البته من هم هکر بوده‌ام و در بخش زبان‌ها در کنار فارسی و انگلیسی و ترکی آذری از سی هم نام می‌بردم ولی حالا کسی که رزومه‌اش را با فایل متن خالص می‌فرستد و زیرش ذکر می‌کند که این رزومه در `emacs` نوشته شده و بالایش یک اسکی آرت هکر گلایدر می‌گذارد را جدی نمی‌گیرم. نه از این نظر که حتما بی‌سواد است بلکه از این نظر که در یک کار تیمی افرادی لازم هستند که با اجتماع اطراف کنار می‌آیند.

> سوء تفاهم نشود! تکست محض و گیک‌بازی و آوردن گیک کد هیچ اشکالی ندارد. چیزی که مشکل دارد به رخ کشیدن این چیزها است.

مرتبط بنویسید

رزومه نباید خیلی طولانی باشد. بعضی‌ها حتی می‌گویند اگر حرفه‌ای هستید بیشتر از یک صفحه ننویسید. اطلاعات مختلف برای موقعیت‌های مختلف ارزش‌های متفاوت دارند. اگر برای یک مرکز تحقیقاتی رزومه می‌فرستید معلوم است که تک تک مقاله‌هایتان مهم است ولی اگر برای یک شرکت برنامه‌نویسی اپلای کرده‌اید، مهمترین بخش رزومه مشارکت‌های شما در پروژه‌ها (و بخصوص پروژه‌های آزاد جهانی) است. من هیچ وقت

صفحات سه و چهار و پنج رزومه‌ام را با فهرست همه جاهایی که سخنرانی کرده ام پر نمی‌کنم و فقط اگر در شغل خاصی لازم باشد ممکن است در متن پاراگراف اول اشاره کنم که در جامعه لینوکس سخنران هستم. اگر تازه از دانشگاه بیرون آمده‌اید اصلا لازم نیست تلاش کنید رزومه‌تان را به دو صفحه برسانید و ذکر تحصیلات و مشارکت‌های احتمالی در پروژه‌ها و حوزه‌های علاقمندی و دانش کافی است. یک رزومه خوب معمولا از یکی دو صفحه بیشتر نیست.

کمی شخصی باشید

و البته معنی سرتیتر قبلی این نیست که قرتی بازی‌های شخصی‌تان را کلا فراموش کنید. علاقمندی‌های شخصی را بنویسید. اگر کار جالبی به ذهنتان می‌رسد انجام دهید ولی یادتان باشد که این‌ها بخشی از ورودی‌هایی هستند که برداشت دیگران از شما را شکل می‌دهند. این مساله هم هست که برای مثال مدیر آی تی لازم است بتواند رزومه شما را به بخش منابع انسانی بفرستد پس موضوعات بیش از حد غیرعرف ممکن است علی‌رغم جالب بودن از نظر شما و حتی مدیر، از نظر مدیر منابع انسانی کارها را سخت کند.

شروع کنید

مهمترین قدم برای داشتن یک رزومه قابل قبول، شروع کردن به ساختن آن است. ما گرایشی داریم که همه چیز را به یک زمان پرفکت موکول کنیم و این کاملا اشتباه است. همین حالا یک ادیتور باز کنید و درفت اول رزومه‌تان را بنویسید. انتخاب ابزار یا یادگیری ابزار جدید اگر بیشتر از پنج دقیقه طول بکشد ضرر است. متن زیر را در یک فایل کپی کنید و قدم اول نوشتن رزومه برداشته شده. انتظار می‌رود با نیم ساعت کار، رزومه قابل قبولی آماده کرده باشید.

```
Name:
Family Name:
Nationality:
Applying for:
=======================================================
A paragraph about me and why I’m applying for this po-
sition.
=======================================================
Work History:
=======================================================
Knowlege Areas:
Programming
(a table)
Technologies
(a table)
Languages
( a table)
=======================================================
Education History:
=======================================================
Some personal touch here. Your hobbies, interests, geek
code, ... whatever you like.
```

۴.۵ آیا شرکت خوبی هست که قدر من رو بدونه

این مطلب مدت ها است که قراره نوشته بشه. چون مدت ها قبل یک دوست خیلی عزیزم ازم پرسیده:

> من سالهاست که وبلاگت رو می خونم و از طرفدارای پر پر و پا قرصشم. طرز نوشتن و بیانت حرف نداره. خیلی وقت ها پیش میاد که بعد از هفته ها یا حتی ماه ها برمیگردم و بعضی از مطالبت رو می خونم. مثل پست «چیزهایی که در مدرسه یاد نمی دهند» که دو باری خونده بودمش و برای بار سوم امروز خوندم ...
> سوالی که برام پیش اومده اینه که آیا واقعا باید هر روز به خودت اون قوانین رو تکرار کنی و بمونی و بجنگی به قیمت به تحلیل رفتن زمان، سلامت روح و جسم و توانایی هایی که می تونی شکوفاشون کنی اما فرصتی برات نیست، یا که نه، شرایط مطلوب تری هم یه جایی ممکنه پیدا بشه؟ تو تجربه ای که در کار با شرکت های خارجی دارم، دستم اومده که همچین جاهایی هستن که قدر بدونن و احترام بذارن و البته به نظرم شاید بی عدالتی هم نباشه اگر وقتی کاری رو درست انجام نمی دی، رفتار خوب دریافت نکنی، اما وقتی کارت به بهترین نحو ممکن انجام می شه، آیا بازم باید رفتار زشت و بد ببینی؟

متاسفانه جواب من خیلی دلگرم کننده نیست. شرکت ها بهتر و بدتر دارن ولی چیزی به اسم شرکت قدردان خوب مهربون باشعور قابل وفاداری دوستانه فلان فلان وجود نداره. دلیلش هم ساده است: سرمایه داری. هدف سرمایه داری شکوفا کردن انسان ها نیست بلکه کسب سود بیشتره.

ما توی عصری توی کره‌ای زندگی می کنیم که سیستم تولیدش سرمایه داری است. شاید کشورها اسم خودشون رو بذارن «جمهوری خلق» یا «جمهوری اسلامی» یا هر چیز دیگه ولی در نهایت چیزی که کشورها و در سطح بالاتری کل سیاره رو می چرخونه سرمایه داری است. سرمایه داری یعنی کسی که سرمایه داره قوانینی رو وضع می کنه برای بیشتر کردن سرمایه اش و اگر هم این وسط یکی دلش برای بقیه بسوزه و سعی کنه مهربونتر باشه یا مثلا خدمات بیشتری بده یا کار کمتری بکشه یا هر چی، سریعا توسط شرکت هایی (یا در سطح کلان بخونین کشورهایی) که این «سوسول بازی»ها رو ندارن خورده می‌شه. مثلا شرکت اروپایی فنلاندی من که توش شخصیت آدم ها مهم بود و انسان ها حق داشتن در ابعادی که دوست دارن توش پیشرفت کنن و سفرهاشون کاملا راحت و نسبتا لوکس باشه و ... سریعا توسط شرکت چینی رقیب که توش همه باید مثل سرباز کار کنن وگرنه با یکی از اون یک میلیارد و دویست میلیون و خورده ای (که رقم یکان اون خورده‌ای اش برابر جمعیت فنلاند است) جایگزین می‌شن، تهدید می شه و مجبوره اون خدماتش رو قطع کنه.

البته نه اینکه همه جا مثل هم باشه و همه جا وحشتناک باشه و اینها. ولی یکسری توهم رو نباید داشته باشیم. مثلا این توهم که در خارج کار راحت تره و توی ایران ما خیلی زحمت می کشیم یا مثلا این توهم ریشه ای تر که می رم یک شرکتی و خیلی مهم می شم و توش به من احترام میذارن و ارزش کارم رو درک می کنن. مشکل این نیست که شرکت ها نفهم هستن، مهم اینه که هدف شرکت ها با هدف ما فرق داره. شرکت

ها دنبال حداکثر کردن سودشون هستن و ابتکار و خلاقیت و ... شما معمولا هنر خیلی بزرگی در این جریان بازی نمی کنه. همونطور که گفتم بعضی شرکت ها بهتر هستن چون بهتر میفهمن که یک کارمند راضی می تونه سود بهتری برسونه یا مثلا شرایط کاری بهتر باعث جذب نیروهای بهتر می شه که در نهایت سود بهتری می دن یا مثلا با شعورترهاشون شروع می کنن به جایگزین کردن سیستم هرمی با سیستم‌های ماتریسی که توش شما یک رییس ندارین که بتونه شما رو ناراحت کنه (دو تا دارین که تا حدی با هم تضاد منافع دارن و در نتیجه شما راحت تر زندگی می کنین).

اینها شرکت ها رو با هم متفاوت می کنن. به قول سوفیا لورن «پول خوشبختی نمی آره ولی من ترجیح می دم توی یک کادیلاک گریه کنم تا توی یک فولکس». شرکت ها هم با هم فرق دارن. شرکت خیلی جذابی وجود نداره که به شما هم حقوق بده هم توش کاملا خوشحال باشین ولی معلومه که بدون شک بعضی شرکت ها از بعضی شرکت ها بهتر یا با شعورتر هستن اما هیچ تضمینی نیست که در دوران بد اقتصادی، لوس بازی‌های اروپایی رو بذارن کنار تا بتونن با رقبایی که مثل ماشین جنگی کار می کنن، رقابت کنن.

توصیه من؟ اگر واقعا دنبال جایی برای خلاقیت یا پیشرفت چند بعدی یا پولدارشدن یا اینجور چیزها هستین حتما برین سراغ شرکت خودتون یا شرکت‌هایی اونقدر کوچیک که با ورود بهش می تونین صاحبش باشین نه حقوق بگیرش. در غیراینصورت امید بیخودی رو قطع کنین و بدونین که هیچ جای دنیا بر اساس سیستم «قدر دانی از کار خوب» کار نمی کنه. اگر واقعا قرار بود شما برین جایی و خوشحال باشین که دیگه بهتون حقوق نمی دادن (: حالا ممکنه بعضی مزایای خوب وجود داشته باشه یا در دوره هایی از چیز یاد گرفتن یا هر چیز دیگه شاد باشین ولی در کل دلیل اینکه این شرکت ها به آدم ها حقوق می دن اینه که کسی حاضر نیست اون کارهاشون رو داوطلبانه انجام بده. امید بیخودی رو قطع کنین و بدونین که با اینکه شرکت ها خوب و بد دارن ولی همه شون بر اساس منطق «سود» کار می کنن و در یک دنیای پر از رقابت.

خودتون رو هم گول نزنین. نمی تونین یک شرکت خوب شاد درست کنین چون شرکت های خشن بیشتر-از-شما-سود-محور بهتون اجازه نفس کشیدن طولانی نخواهند داد.

اما چاره چیه؟ توصیه بودایی ها رو جدی بگیرین «خواستن رنج است». اگر با این دید می رین به یک شرکت که خیلی باشعور باشه، اشتباه میکنین و چیزی رو می خواین که باعث رنج خواهد شد. در مقابل بدونین که وقتی به جایی می رین برای منافع خودتون رفتین (از چیز یاد گرفتن تا سفر رفتن تا حقوق سر ماه تا ...) و به توصیه اولین مدیرخط من آقای شهپر هم گوش بدین: «غر نزنین خودتون رو هم با بقیه مقایسه نکنین. تا لحظه ای که حس می کنین کار کردن در اینجا راتون می صرفه اینجا باشین و اگر فکر کردین به نفعتون نیست، برین. خیلی ساده». موقع پیدا کردن شرکت به چیزهایی که واقعا براتون مهمه (مثلا برای من سفر، دوست خوب، محیط کار جالب، رزومه جذاب در آینده و ... مهمتر از حتی حقوق است) توجه کنین و سعی کنین به جایی برین که برای آینده به دردتون بخوره. حس نکنین که بهترین جا رو پیدا کردین و گول زرق و برق سرمایه داری رو نخورین چون پشت ظاهر جذاب بهترین شرکت‌ها، منطق خشک و خشن سود است که

کار می کنه[۱].

[۱] برای بحث به اینجا مراجعه کنین jadi.net

۵.۵ انتخاب مسیر حرفه ای

فرض کنید یک نفر بیاید و به شما بگوید **متخصص کامپیوتر** است. چه برداشتی می کنید؟ متخصص شبکه است؟ متخصص سیستم عامل است؟ بلد است چطور باید یک دیتابیس خاص را برای بالاترین پرفرمنس تنظیم رکرد؟ برنامه نویس است؟ معمار نرم افزار است؟ ممکن است جواب هیچکدام باشد و اصولا این آدم به شکلی دقیق تر به شما بگوید که **متخصص امنیت** است. راستش را بخواهید هنوز هم نمی شود گفت این آدم در دنیای بزرگ آی تی دقیقا چکاره است.

یک متخصص امنیت ممکن است بلد باشد یک سیستم عامل را امن کند و ممکن است مهارتش در تنظیم کردن سخت‌افزارهایی مانند فایروال‌ها باشد. حتی شاید لازم باشد در این مرحله بپرسیم که تخصص دوستمان در کدام سری ابزارهای امنیتی است یا کدام خانواده از سیستم عامل‌ها. این فرض سختی نیست چون کافی است ما به گذشته این آدم (در قالب رزومه یا سوابق کار یا تحصیلات یا علایقی که رویشان زمان گذاشته) نگاه کنیم و بالاخره خواهیم توانست جواب را بیابیم.

اما اگر فلش زمان برعکس باشد چه؟ نگاه به گذشته آسان است ولی چه می‌شود اگر بخواهید به آینده نگاه کنید و بگویید شمایی که **آی تی** را دوست دارید، در کدام شاخه قرار است متخصص شوید؟ این فصل از کتاب تلاش می کند به این سوال جواب بدهد و برای اینکه هم خودم و هم شما را راحت کرده باشیم؛ جواب نهایی را همین اول می دهم: هیچ کس نمی‌داند و تا متخصص نشده‌اید، هیچ کس نخواهد توانست پیش بینی کند چکاره خواهید شد و بهتر است خودتان هم ندانید... چرا؟ به خاطر بایاس پروجکشن!

در این مقاله این بایاس را بررسی کرده، با نگاه به چند موضوع مهم بهترین پیشنهاد ممکن از نظر من در جواب به اینکه «کدوم خط از آی تی رو بخونم» را می‌دهیم.

بایاس پروجکشن

انسان‌ها بر خلاف تبلیغاتی که به راه انداخته‌اند، چندان هم عقلانی و هوشمند و منطقی نیستند. ما ادای منطقی بودن را در میاوریم ولی مغزمان پر از مکانیزم‌های دفاعی است که تلاش می کند دنیا را ساده‌تر و ناامن‌تر از چیزی که هست تصور کند تا بتواند خودش را زنده نگه‌دارد.

بایاس‌های رفتاری ما فهرستی بسیار طولانی دارند؛ مثلا اینکه در صورت ادعای احتمالا دروغ نسبت به یک خطر آنی، کسانی که دروغ را می‌پذیرند زندگی ای طولانی تر از کسانی دارند که سعی می کنند با تفکر انتقادی جواب واقعی را کشف کنند.

دقیقا همان مثال که اگر کسی به شما «دیدم که در کفشی که دم در گذاشته‌ای یک مار مخفی شده، من می توانم برایت کفش را دور بیاندازم» به نفع شما است که به او گوش کنید و این خطر را به جان بخرید که کلا دروغ گفته باشد و کفش شما را برای خودش بردارد تا اینکه شخصا بروید و بررسی کنید که آیا داخل کفش یک مار وجود دارد یا نه. این بایاس (تمایل، تعصب،...) عامل به وجود آمدن بسیاری عقاید است که سعی می

کند از ما در برابر طبیعت وحشی حفاظت کند (: اما به انتخاب شغل ما ارتباط چندانی ندارد.

اما بایاسی ادراکی ای که در اینجا به ما مربوط است، بایاسی است به نام بایاس پروجکشن:

این اشتباه ادراکی که ما فکر می کنیم در آینده هنوز همین آدمی هستیم که حالا هستیم.

یک نمونه بسیار ساده ولی بسیار واضح از این بایاس زمانی است که به خودمان می‌گوییم «از فردا ورزش می کنم / درس می خوانم / خوب غذا می خورم و ...». در لحظه برگشتن از شرکت یا دانشگاه یا مدرسه، فکر می کنیم که ما فلان کار را دوست داریم پس از فردا آنکار را خواهیم کرد. ما در اینجا درگیر بایاسی هستیم به نام **پروجکشن بایاس** که شاید بتواند آن را تمایل به انداختن تصویر حال به اینده ترجمه کرد.

ما در این لحظه فکر می کنیم که فردا دقیقا همین آدم با همین علایق خواهیم بود و معلوم است که دوست خواهیم داشت ورزش کنیم یا درس بخوانیم یا هر چیز دیگر ولی وقتی فردا از راه می‌رسد، آدم دیگری هستیم تحت تاثیر هورمون‌ها، خستگی‌ها، تفکرات و درگیری‌های آن زمان که احتمالا اولویت درس خواندن یا سالم غذا خوردن یا ورزش کردن را کم می‌کنند (:

انسان در هر لحظه ساخته می‌شود و اشتباه‌ترین فکر این است که در شانزده سالگی فکر کنیم می دانیم قرار است در سی سالگی دقیقا چه کاره باشیم و حالا از طریق یک خط مستقیم باید به آنجا برسیم.

در متن‌هایی که سعی می کنند با احساسات شما بازی کنند تا کتاب‌هایشان را بفروشند، از شما می‌پرسند «آیا به آرزوی کودکی خود رسیده‌اید؟» و بعد از کمی بازی احساسی از شما می‌خواهند قهرمان باشید و به باورهای کودکی برگردید و ... و تنها هدفشان هم تحریک احساسات است تا فکر کنید «بله چقدر زندگی من به فنا رفته است» و از اینکه پول کتاب یا سخنرانی یا کلاس را داده‌اید، احساس مثبتی بکنید.

بدون شک خودتان هم این را می‌دانید که هیچ کس بعد از بیرون آمدن از این جلسه‌ها، کار مهندسی اش را رها نمی‌کند تا فضانورد یا غواص بشود!

عمیقا پیشنهاد می کنم که در هر لحظه از زندگی باید در حال فکر کردن به زندگی‌تان باشید و کاری که می‌کنید و نگران این باشید که نکند روزمره شوید و ... ولی همانقدر که خنده‌دار است یک مرد چهل ساله، کارش را رها کند و از یک جوان پانزده ساله بپرسید «حالا شما بفرمایید من چکاره بشوم» این هم خنده دار است که یک نفر در پانزده سالگی بخواهد به طور دقیق مشخص کند که در چهل سالگی قرار است مشغول چه کاری باشد.

اگر هنرپیشه نشوید، شغل هالیوودی نخواهید داشت

من در نوجوانی عاشق فیلم هکرها [1] و عاشق هکر شدن! تجربه‌های اول هک بسیار هیجان انگیز بود ولی سریعا کشف کردم که آخر راه مشکلات قانونی خواهد بود و دردسرهایی که هیچ کس در دنیای واقعی آن‌ها را نمی‌خواهد. همزمان عاشق کدنویسی هم بودم ولی خیلی زود کشف کردم که اگر شغل کسی کد نویسی باشد

[1] www.imdb.com

وظایف روزانه‌اش اینهاست:

- امروز یک تابع بنویس که با گرفتن یک رشته، مجموع ستون یازدهم همه را برگرداند. اعداد ممکن است بالای پانزده رقم داشته باشند
- یک تابع بنویس که به شبکه شتاب وصل شود - درست است که ای پی آی غیرجذاب است ولی همین است که هست
- یک فرم آنلاین داریم که کاربر خواسته خروجی اکسل داشته باشد. امکان دانلود اکسل را درست کن
- در دانلود اکسلی که دیروز درست کرده بودی، حروف فارسی علامت سوال دیده می شود. این را اصلاح کن
- کاربر اکسل ۲۰۱۲ دارد ولی تو فایل اکسل ۲۰۱۰ می دهی، این را اصلاح کن
- تیتر اکسل‌ها را بولد کن
- ...

هنوز به نظر من هم جالب است ولی قول می دهم بعد از چهار سال کار تکراری، کمتر کسی هنوز مدعی می شود که «کدنویسی» را دوست دارد. اگر می‌خواهید در مورد شغل‌های مختلف اطلاعات کسب کنید، با آدم‌ها واقعی حرف بزنید و شرح شغل‌های واقعی را ببینید. متخصصین امنیت این روزها پچ نصب می کنند، پروسه‌های استاندارد امنیتی را در سازمان می نویسند و پیاده می کنند و روی فایروال‌ها چند دستور تکراری تایپ می‌کنند. این کاملا با فیلم هکر که در آن یوجین در چنین جایی با هکرها می‌جنگید فرق دارد.

و همین دو هفته پیش بود که ما تلاش می‌کردیم یک هکر پانزده ساله که سعی کرده بود به ما حمله کند و توسط پلیس فتا دستگیر شده بود را به زندان نفرستند.

در واقعیت همه کارها تا حدی حوصله سربر هستند چون در غیراینصورت کسی برای انجامشان به کسی حقوق نمی داد. من ساعت‌هایی از روز در نقش تحلیلگر سیستم در حال باز کردن فایل‌های `csv` و ذخیره آن‌ها به شکل `xlsx` هستم تا بخش مالی فشارش را روی بخش آی.تی. کم کند و همکارم همزمان دارد ستون‌های اکسل را چک می‌کند که جمعشان با جمعی که توسط بخش مالی محاسبه شده بخواند و سیستم‌های ما را دچار مشکل نکند - تغییر دائمی فایل‌ها از طرف بانک باعث شده هنوز فرصت نکرده باشیم این کارها را اتوماتیک کنیم. شما هم وارد هر بخشی از دنیای آی تی بشوید، باید با این کم و کاستی‌ها سر و کله بزنید و هیچ وقت فراموش نکنید که هیچ شرکتی نیست که به طور کامل قدر شما را بداند.

متخصص شدن یک مسیر تک خطی نیست

مساله مهم بعدی این است که اصولا **پیشرفت** یک مسیر مستقیم نیست. درک خوب از جوانب مختلف موضوع چیزی است که یک متخصص آی تی را از یک آدم تک بعدی جدا می‌کند. بدون شک جامعه بسیار تخصصی شده و کارها تفکیک و در نهایت شما باید متخصص یک یا دو یا سه چیز باشید ولی داشتن دانشی وسیع در مورد مفاهیم مرتبط به راحتی می تواند شما را از خیل عظیم فارغ التحصیل‌ها جدا کند. به مساله اینطور نگاه کنید که جنگلی به اسم آی تی در جلوی شما است و در یک رقابت قرار است شما در کنار صد نفر دیگر جنبه‌هایی از این جنگل را **بشناسید** و از طرف دیگرش بیرون بیایید.

نکته تعیین کننده این است که در نهایت برای اینکار چندین سال وقت دارید! در این رقابت برنده کسی نیست که با حداکثر سرعت در یک مسیر مستقیم از یک طرف جنگل وارد می‌شود، مسیر بسیار مستقیمی را یاد می گیرد (مثلا سی، سی پلاس پلاس، جاوا، پایتون) و از آنطرف خارج می‌شود و دو سه سالی که اضافه آورده را صرف تقویت آن مهارت‌ها می‌کند یا بیکار گوشه‌ای می‌نشیند تا دوستانش هم برسند.

برنده این رقابت کسی است که با حوصله در جنگل قدم می‌زند و حتی اگر مطمئن شد که قرار است از مسیر برنامه نویسی خارج شود، خودش را با گونه‌های گیاهی دیگر مثل دیتابیس‌ها و سیستم عامل‌ها هم آشنا می‌کند و حتی مقدار قابل قبولی وقت می‌گذارد تا به گم شده‌های جنگل هم کمک کند و هم کار تیمی یاد بگیرد هم برای خودش اسم و رسمی به هم بزند. این زیگ زاگ رفتن در جنگل تلف کردن وقت نیست بلکه

پیدا کردن دانشی عمیق تر به مسیری است که بقیه سعی می‌کند به سرعت طی کنند[1].

چنین کسی را در دنیای کامپیوتر `Full Stack` می‌نامند: کسی که از همه بخش‌های مورد استفاده

[1] نکته: در این تصویر برنامه نویس ها مرد هستن و کسایی که در مورد پارتنر برنامه نویسشون حرف می زنن زن (: دنیای واقعی فرق داره و بخصوص توی کشور ما خانم ها حضوری خیلی پر رنگ و قوی توی دنیای آی تی دارن. مواظب باشین مغزتون با تصاویر رسانه‌ای جهت دهی نشه

اش حداقل سر در می‌آورد و اگر برنامه ای می نویسند کاملا درک می‌کند که سیستم و عامل و دیتابیس و وب سرور چه نقشی در اجرای آن ایفا خواهند و شبکه چگونه در مقابل درخواست‌ها عکس العمل نشان خواهد داد. توجه کنید که در اینجا بحث این نیست که حتما لازم است یک نفر از همه چیز سر در بیارد! من طرفدار تخصص هستم ولی نکته این است که زیگزاگ رفتن برای کشف علاقه، فوایدی بسیار زیاد دارد و در نهایت مفیدتر از طی کردن یک خط مستقیم برای رسیدن به آخر راه. زندگی طولانی است و به هر «آخر»ی که دیرتر برسید، برنده هستید (:

پس بهترین کار چیست؟

در قدم اول مهمترین نکته در تمام زندگی این است که:

آسانسور پیشرفت هنوز اختراع نشده و اگر می‌خواهیم پیشرفت کنیم، باید قدم به قدم پله‌ها را طی کنیم.

و در قدم دوم تلاش برای کشف چیزی که از آن لذت می‌بریم و بهترین شدن در آن.

متاسفانه در جامعه بیمار ما، **نمایش** ارزش بیشتری از دانش واقعی پیدا کرده و اکثرا علاقمند هستیم خیلی سریع کشف شویم یا تعجب آدم‌ها را نسبت به خفن بودن خودمان بربیانگیزیم. این خطرناکترین چیز در پیشرفت واقعی است چون باعث می شود **انسان از خودش جلو بیافتد** . فرض کن من دوست داشته باشم موشکی بسازم که به ماه می رود و شش ماه در این مورد فانتزی سازی کنم و بعد از شش ماه با کمی عکس ساختگی به دوستانم بگویم که سوختی اختراع کرده ام که توانسته یک بطری نوشابه را به لبه جو بفرستد!

در دیوانه خانه ای که ما در آن زندگی می‌کنیم حتی بعید نیست روزنامه ها از من تقدیر کنند و بگویند «یک جوان ایرانی توانسته در زیرزمین خانه‌اش سوخت موشکی بسازد بهتر از ناسا» و اصلا بعید نیست که برای ادامه تحقیقات بودجه هم بگیرم.

مثلا از دفتر فلان مرجع که دو سه سال پیش به سازمان استعدادهای درخشان نامه ای نوشته بود که چرا از جوانی که سیستم ضد جاذبه ای ساخته که می‌شود با آن بشقاب پرنده ساخت حمایت نکرده‌اند. در این شرایط من دیگر هیچ وقت نخواهم نشست مثل آدم شیمی پایه بخوانم و چند تجربه کوچک کنم و با یک بالن دوربینی عکاسی به لایه‌های بالای جو بفرستم و بعد با جی پی اس دوربین را پیدا کنم و عکس‌هایم از کره زمین را ببینم.

دقیقا همین مساله در پیشرفت آی تی هم مشهود است. یاد گرفتن برنامه نویسی، شبکه، امنیت و هر چیز دیگری سخت است و ما در همان اول دوست داریم بین دوستانمان هکر بزرگ، برنامه نویس کرنل، نفوذ کننده به ناسا و دست چپ لینوس تورووالدز در پذیرش پچ‌های زمانبندی داخلی کرنل باشیم. راز اینکه بتوانیم در ده سال بعد واقعا چیزی باشیم که می خواهیم، لذت بردن از مسیر و یادآوری روزانه این واقعیت است که **آسانسور پیشرفت هنوز اختراع نشده** .

اگر می‌خواهید مسایل بالا را جمع بندی کنید و هم مسیر زیگ زاگ و کشف جنگل آی تی را داشته باشید

و هم پیشرفت قدم به قدم را، شاید بهترین کار این است که **در هر لحظه از هر کاری که می‌کنید لذت ببرید** . ببینید در هر دوره ای به چه چیزی علاقه دارید و قدم به قدم و با حوصله از پایه آن را یاد بگیرید. منطقا مسیر می‌تواند شامل اینها باشد و هر چیز دیگری که شما از آن لذت ببرید:

- مقدمات شبکه شامل درک مفاهیم آی پی و ساب‌نت و روتینگ و سوییچینگ مقدماتی
- درک پروتکل‌های اینترنتی مانند TCP/IP و لایه بندی شبکه و بعد رفتن سراغ پروتکل‌هایی مانند اف تی پی، اچ تی تی پی، اس اس اچ
- آشنایی با ابزارهای شبکه مانند دستورات سیسکو یا جونیپر و دوره‌هایی مانند CCNA
- آشنایی با سیستم عامل‌هایی مانند گنو/لینوکس به شکل دسکتاپ و بعد سرور و یاد گرفتن ابزارهای مربوط به آن‌ها
- اشنایی با تکنولوژی های وب مانند اچ تی ام ال و سی اس اس و زبان‌های برنامه‌نویسی ای مانند پی اچ پی به همراه مای اسکویل
- یاد گرفتن خوب یک زبان برنامه نویسی پایه مانند سی و بعد سی پلاس پلاس یا جاوا
- حرفه ای شدن در یک زبان اسکریپتی مانند پایتون یا نود.جی.اس.
- آشنایی با بانک‌های اطلاعاتی غیراسکوئل مانند مونگو یا کوچ
- آشنا شدن با معماری نرم افزار، متودولوژی‌ها، چارچوب‌ها و استانداردها. چیزهایی مثل RUP، Cobit، ITIL، Agile، SCRUM
- مشارکت در پروژه‌های آزاد برای یاد گرفتن شیوه کار تیمی و همکاری با دیگران و ارتباط با جامعه
- یاد گرفتن ابزارهایی که یک توسعه دهنده از آن‌ها استفاده می‌کند مانند git و eclipse
- لذت بردن از تاریخچه‌ها، زندگی‌نامه‌ها، سرگذشت‌ها و اخبار و مفاهیم فلسفی گیک‌ها

کافی است شما از کارهایی که می‌کنید لذت ببرید تا هر روز با علاقه چیزهای جدید یابد بگیرید. مطمئن باشید که چیزی را بدون فهمیدن و فقط برای نمایش نتیجه کپی پیست نمی کنید و بدانید که اگر هر چیزی که انجام می دهید را یاد بگیرید، خیلی خیلی زود محصولاتی را خواهید ساخت که دیگران را به تعجب بیاندازد. یک دفترچه برای ایده‌ها داشته باشید و هر بار که می خواهید سراغ موضوع جدیدی بروید بررسی کنید که

آیا دلیلش واقعا علاقه به چیزی جدیدی است یا فرار کردن از چیزهای قبلی وقتی که کمی سخت / جدی می شوند.

در این راه محصولاتتان را به نمایش بگذارید و با علاقه به نقد بقیه گوش بدهید. دقت کنید که وظیفه دیگران کشف هنرهای شما نیست بلکه اگر کاری که می کنید به اندازه کافی خوب باشد آرام آرام دیده خواهید شد. استقامت داشته باشید و یادتان باشد هکرها و گیک‌ها از کارهایی که می کنند لذت می برند نه از نتیجه ای که می توانند به دیگران نشان دهند.

شما هم بدون ترس چیزهایی را یاد بگیرید که دوست دارید و کارهایی را بکنید که در طولانی مدت به نفعتان است و حواستان باشد که زندگی چه بخواهید چه نخواهید بسیار بسیار طولانی است! ما شاید تا هجده سالگی اصولا به شکل مستقل زندگی نکنیم و بعید است کسی دوازده سالگی اش را هم زندگی مستقل بشمرد. پس یک آدم بیست و پنج ساله شاید فقط پنج یا حداکثر ده سال واقعا به این معنا که تصمیم مستقل می گیرد و خودش تعیین می کند چه کارهایی بکند و چه کارهایی نکند **زندگی کرده باشد** ولی چیزی حدود پنجاه سال هنوز از عمرش باقی مانده! با این تفسیر خیلی نگران **دیر شدن** نباشید.

راستش این است که وقتی کشف کردیم چه کاره هستیم و آن کاره شدیم دیگر چیز زیادی از یک گیک باقی نمی‌ماند پس اتفاقا توصیه اصلی این است که تا جایی که می‌توانید رسیدن به آخر خط را عقب بیاندازید و از گشت زدن در جنگل لذت ببرید - اما به شرط اینکه کلا نخوابید (:

۶.۵ دیتاسنترها

من اولین کار جدی‌ام رو که قبول کردم، یک دلیل داشت: ممکن بود بعضی روزها توی دیتا سنتر کار کنم. فکر اینکه خواهم تونست کنار سرورهای سان، آی بی ام و اچ پی باشم و در کنار لینوکس‌های خودم با سیستم‌عامل‌هایی مثل سولاریس، اچ پی یو ایکس و غیره کار کنم به نظرم فوق العاده بود. دفعات اول فضا برام خیلی عجیب بود. کیسه‌هایی که دور کفش می‌کشیدیم، رک‌های بزرگ، کی وی ام ها و صدای دائمی wooooVVVVV کامپیوترها که بعد از بسته شدن در دولایه تنها چیزی بود که توی گوش می‌چرخید.

اما همیشه فضا اینطوری نموند. راه دور، سرما، ممنوع بودن خوردن و آشامیدن و آنتن ندادن موبایل‌ها خیلی زود شروع کردن ناراحت کننده شدن و بعد از یکسال منم مثل بقیه تیم نه فقط مشتاق رفتن به بالا سر سرورها نبودم که همه تلاش و اصرارم رو داشتم که باید با وی.پی.ان. داشته باشم و از راه دور بتونم به سرورها وصل بشم. وقتی هم به کشورهای دیگه سفر کردم دیدم در بسیاری از کشورهایی که سخت‌گیری کمتری روی کیفیت اجرای پروژه هست، تیم‌های اجرایی با گذاشتن یک دونه لپ تاپ در رک مورد نظر و اجرا کردن برنامه‌هایی مثل تیم‌ویور اصولا بدون سر و کله زدن با مشتری، از راه دور به سیستم‌هاشون وصل می‌شن.

اینکار مشکل امنیتی داره! هیچ وقت انجامش ندین مگر اینکه مشتری تاییدش کرده باشه.

نتیجه‌اینه که کار کردن توی دیتاسنتر ترکیبی است نکته‌های مثبت و منفی. در حوزه منفی، از سرما و صدا و گشتن به دنبال صندلی برای نشستن و کابل برای وصل شدن به سرور. صدای فن کامپیوتر خودتون رو صد برابر بیشتر کنین و کولر روی ۱۲ درجه تنظیم کنین و جلوش بشینین تا بتونین درکی از شرایط دیتاسنتر داشته باشین. در مقابل حوزه مثبت اینجاست که خودتون تصمیم می‌گیرین چیکار کنین، مدیرها بالا سرتون نیستن، معمولا تنها هستین یا با تیم خوبی کار می کنین و به کامپیوترهای جالبی دسترسی دارین و تجربه‌ای رو می‌کنین که عکس‌هاش و خاطره‌هاش آب از دهن هر تازه‌واردی سرازیر می‌کنه.

ولی برای چنین تجربه‌ای نیاز به چه مهارت‌هایی دارین؟ شاید براتون جالب باشه که مهارت‌های غیرفنی حتی مهمتر از مهارت‌های فنی هستن. بذارین مرور کنیم.

قدرت شخصیت و اعتماد به نفس

توی دیتاسنتر شما تنها هستین و خیلی وقت‌ها لازمه بتونین شخصا تصمیم بگیرین و اجرا کنین. حتی خیلی وقت‌ها ممکنه داخل دیتاسنتر به اینترنت هم دسترسی نداشته باشین (واقعا! موبایل هم گاهی توی کانکس‌های بزرگ دیتاسنترها یا زیر زمین آنتن نمی‌دن) و در نتیجه باید داکیومنت و سواد و غیره رو بریزین روی لپ‌تاپ تا کاملا خودکفا بشین. راستش رو بگم من حتی در جاهایی کار کردم که اجازه نداشتین لپ‌تاپ و موبایل هم توی دیتا سنتر ببرین (:

همزمان بحث‌های روانی هم مطرحه. یک نفر ممکنه به راحتی در یک محیط پر از نویز سفید و باد سرد و دیوارهای روشن دچار توهم بشه یا نتونه به شکل درستی تمرکز کنه. چنین آدمی مطمئنا به درد کار توی

دیتاسنتر نمی‌خوره.

مهارت‌های فنی

منطقا اگر قراره کسی به شما حقوق بده که برین توی یک دیتاسنتر، قراره یک کاری اونجا بکنین. ممکنه متخصص شبکه باشین، ممکنه تو کار کابل‌کشی باشین (که اتفاقا بسیار جالبه و بانمکه) یا ممکنه سیستم‌عامل بلد باشین. حوزه‌های خیلی خیلی زیادن. شاید شما متخصص یک نرم افزار خاص باشین که قراره نصب یا ساپورت بشه، ممکنه سخت افزار بلد باشین و ممکنه اطلاعات زیادی در مورد سخت‌افزارهایی مثل سرورها، استوریج‌ها (هاردها) و این تیپ چیزها داشته باشین.

اگر بحث مدرک است مدارک شبکه و چیزهایی مثل LPIC می‌تونن شما رو وارد دیتاسنترها کنن و البته دوره‌ها و مدارک طراحی دیتاسنتر چیزی هستن که کلا بقیه زندگی شما رو وابسته به دیتاسنتر خواهند کرد.

اگر از من می‌شنوین سراغ شغل‌هایی برین که باعث می‌شه گاهی توی دیتاسنترها باشین و گاهی بیرون تا حق انتخابتون حفظ بشه. خوندن دیتاسنتر دیزان چیزی شبیه به خوندن کاپیتانی کشتی است؛ توصیه می‌شه قبل از خوندن حداقل یک بار سوار کشتی شده باشین (:

قدرت فیزیکی

کسی که در دیتا سنتر کار می کنه معمولا لازمه صندلی‌اش رو خودش جابجا کنه، کمی کابل بکشه، در رک رو باز کنه - که گاهی سنگینه و حتی لازم می‌شه سروری رو از رک بیرون بکشه یا کی.وی.ام (دستگاه اتصال به سرورها) رو بیرون بیاره و درش رو باز کنه. شاید کلیت کار گاهی تفاوتی با کار در یک خط تولید نداشته باشه. بحث‌های سرما و خشکی هوا هم هست و منطقا اگر مدتی توی دیتاسنتر باشین، احساس گلودرد اصلا غیرعادی نخواهد بود. از اونطرف دیتاسنترها معمولا توی جاهای دورتر از مرکز شهر هستن و باید حوصله رفت و آمد رو هم داشته باشین.

اطلاعات اولیه از دیتاسنتر

خب این رو اکثرا تا وقتی وارد دیتاسنتر نمی‌شیم نداریم اما دوره‌هایی هستن که این چیزها رو آموزش می‌دن. شاید توی همین مقاله هم بعضی عبارت‌ها برای شما کمی عجیب بودن ولی در مراجعه دوم و سوم به دیتاسنتر، خیلی از اونها رو یاد خواهید گرفت. چنین دوره‌های رایگان آنلاینی [1] سعی می‌کنن بخشی از این چیزها رو به شما آموزش بدن.

تجربه‌ای دارین یا لازمه چیزی اضافه بشه؟ به `jadijadi` روی جیمیل خبر بدین (:

[1] `www2.schneider-electric.com`

۷.۵ چگونه در انگلیسی پیشرفت کنیم

واقعیت اینه که همه باید انگلیسی رو در حد معقولی بلد باشیم. کسی که نتونه انگلیسی بخونه در واقع عضوی از جهان نیست و کسی که نتونه در حد مینیمم هم که شده انگلیسی بنویسه (مثلا جواب یک نامه) دامنه تاثیرگذاری و امکان کسب درآمدش محدود به ایرانه.

اما چطوری انگلیسی خودمون رو تقویت کنیم؟ من همیشه می گم با نترسیدن!

```
We study English at least for 6 years before getting our
school diploma but still we are afraid of using it.
```

متن انگلیسی رو خوندید؟ این همون نترسیدن است که می گفتم. ما خیلی وقت ها اصولا بخش های انگلیسی رو نمی خونیم و ازشون رد می شیم و معلومه که هی بیشتر و بیشتر از چیزی که نمی شناسیمش می ترسیم و ازش فرار می کنیم. برای یاد گرفتن انگلیسی باید شجاع باشین و اعتماد کنید به اینکه حداقل شش سال هفته ای دو ساعت انگلیسی خوندین. حالا هر چقدر هم شکسته بسته و با پیچوندن.

حداقل در مورد من یکی از مفیدترین بخش های یاد گرفتن انگلیسی بازی‌های کامپیوتری بودن و بعد اینترنت. اگر تکنولوژی دوست دارین از همین حالا شروع کنین هر روز به صفحه اسلش دات[۱] سر بزنین و اگر نه جاهایی مثل صفحه اول یاهو رو هر روز نگاه کنین و خبرها رو بخونین و رو چیزهایی که ازش خوشتون می یاد کلیک کنین. یک جای دیگه جذاب ناین گگ[۲] است به شرطی که تیترها و گاهی کامنت ها رو هم بخونین و بخندین. اینجوری شروع می کنین به مصرف انگلیسی و اون ترس از بین می ره.

قدم بعدی خوندن یک چیز واقعی به انگلیسی است. داستان های اروتیک کشش دارن ولی من با داستان‌های ترسناک شروع کردم. اولین متن انگلیسی که من کامل خوندمش به شکل غریبی با خوندن پو خرسه[۳] شروع کردم و بعدش با دراکولا[۴] ادامه دادم که به شکل عجیبی تونستم بخونمش و ازش لذت بردم. در خوندن یک داستان باید جذب داستان بشین و خودتون رو درگیر یاد گرفتن تک تک کلمات نکنین.

دوست دیگه‌ای هم داشتم که روش یاد گرفتن زبان براش این بود که از یک جایی خودش رو در اون زبان غرق می کرد، اخبار رو به اون زبان می خوند، فیلم رو به این زبان می دید و سرچهاش رو به اون زبان می کرد. معلومه که اینکار شما رو کند می کنه ولی در عوض شما رو مجبور می کنه که زبان رو یاد بگیرین برای زنده موندن.

و خب فیلم دیدن هم که هست. کسانی که کارشون اینه می گن یکبار بدون زیرنویس فیلم رو ببینین، یکبار با زیرنویس انگلیسی ببینین و دوباره بدون زیرنویس تا بهترین یادگیری رو داشته باشین ولی من که

---

[۱] slashdot.org
[۲] 9gag.com
[۳] Winnie the Pooh
[۴] Dracula by Bram Stoker

حوصله چنین کاری ندارم و هدفم هم این نیست، یکبار بدون زیرنویس می بینم و خلاص.

نکته حساس اینه که باید حواسمون باشه که تقریبا شبیه هر مهارت دیگه ای یک نقطه خاص نیست که شما بگین «من الان دیگه می تونم بفهمم پس دیگه راحتم» بلکه یادگیری ذره ذره اتفاق می افته. شما اگر هدفتون یاد گرفتن است خب باید کمی هم سختی بکشین، کتابی رو کمتر بفهمین، فیلم رو دقیقتر ببینین ولی کمتر بفهمین و ... تا بالاخره پیشرفت کنین. درست مثل یاد گرفتن تایپ سریع.

ممکنه شما الان با نگاه کردن به کیبورد و دو سه تا انگشت کجا و کوله بتونین با سرعتی مثلا ۴۰ کلمه در دقیقه تایپ کنین و سوییچ کردن به تایپ سریع یکهو باعث بشه که سرعت تایپ شما به پنج کلمه در دقیقه سقوط کنه. اما تا قبول نکنین که این کم شدن سرعت رو بپذیرین و درست تایپ کنین، نخواهید تونست پیشرفت کنین. پذیرفتن این کند شدن مقطعی و تلاش بیشتر برای کاری که قبلا ساده تر بوده است که باعث می شه بتونین زبان یا تایپ یاد بگیرین.

پس کتاب و داستان های جذاب بخونین (اروتیک، ترسناک، پلیسی یا هر چی که دوست دارین) و در فضایی انگلیسی باشین (به جای ترجمه های تکنولوزی در سایت های ایرانی، وایرد[۱] و اسلش دات[۲] و آرس تکنیکا[۳] و ...) و حوصله کنین و بدون شک تا حد معقولی زبان رو یاد خواهید گرفت. یادتون باشه که بدون دونستن زبان تقریبا هیچ مهارتی شما رو به آدم متخصص در سطح جهان تبدیل نخواهد کرد.

---

[۱] wired.com
[۲] slashdot.org
[۳] arstechnica.com

۸.۵ آیا خواهید تونست، بدون تخصص، از اینترنت درآمد کسب کنید؟

اینترنت پر است از آدم و پر است از وعده‌هایی که سعی می کنن این آدم ها رو جذب کنن. یک سایت می گه اگر توش ثبت نام کنین دو دقیقه بعد مناسب‌ترین پارتنر رو به شما تحویل می ده، یکی می گه پر استروشهای هک و یکی می گه وارد شدن بهش کافیه که در یک ماه پولدار بشین. اما آیا معقوله ما هم دنبال آرزوهامون وارد این سایت ها بشیم؟ بخصوص توی آخری!

خلاصه‌ترین جواب من اینه که «پول الکی در هیچ جا نریخته». کشوری مثل هند پر است از آدم هایی که اینترنت بسیار بهتری از ما دارن و اگر صد دلار در ماه در بیارن خیلی هم خوشحالن و هم انگلیسی شون از من و شما بهتره و هم وقت بیشتری برای تلاش برای در آوردن این پول دارن. پس اگر روشی باشه که شما واردش بشین و بتونین به درآمد های میلیونی برسین، احتمالا اون آدم ها قبلا پرش کردن. اما اینطوری هم نیست که در اینکارها واقعا هیچ پولی نباشه. بذارین ببینیم از کجاها ممکنه پول در بیاد:

اسپم فرستادن

شما می تونین روی تبلیغات شرکت‌ها کلیک کنین، اسپم بفرستین، کامنت اسپم بذارین، آدم ها رو فالو کنین، اکانت فیک بسازین، کپچا وارد کنین و غیره و غیره و از این طریق پول خیلی کمی به دست بیارین. باید دقت کنین که اینکار شدیدا وابسته به سرعت و هزینه اینترنت شماست و ممکنه در عمل با اینترنتی که ما داریم حتی پول خودش رو هم به زحمت در بیاره. اگر هم به درآمد برسین چیزی بیشتر از مثلا پنجاه دلار در ماه نخواهد بود.

اسپم بودن

احتمالا پر درآمدترین «کار» اینترنت فعلا اسپم بودن است. شما می تونین مطالب وبلاگ بقیه رو بدزدین و تو وبلاگ «خودتون» پست کنین. فول آلبوم فلان خواننده رو لینک بدین و الکی هر چی لغت مورد علاقه مردم به ذهنتون می رسه رو توی بخش تگ ها بزنین و یک پولی هم بدین که با چند تا باکس اسپمر تبادل لینک کنین و رتبه بگیرین و بعد تبلیغ های بزرگ کننده حلزون و عینک ضد ترشح و کرم باکس تبلیغ کنن (: این روش اتفاقا می تونه شما رو به درآمد قابل قبولی برسونه.

مثلا در رنج حتی یک تومن در ماه ولی نکته اینه که تقریبا بیست و چهار ساعته باید تلاش کنین اینترنت رو کثیف کنین و مطلب بدزدین و سه روز هم اگر کار نکنین درآمد تقریبا قطع می شه در حالی که اگر نصف همین تلاش رو بذارین روی یک کار شرافتمندانه، میتونین به راحتی مثلا ساعتی پنجاه تومن درآمد داشته باشین - به عنوان یک سیستم ادمین خوب. معادل این تیپ کارها در دنیای بیرون اینترنت، شمع سازی و

پرورش قارچ در خونه است.

کارهای تبلیغاتی جهانی

موارد غیرکلاهبردار در جهان هم هستن که احتمالا اگر به این حوزه علاقمند هستین بهترین گزینه برای شماست. مشهورترین پول دهنده در جهان گوگل و سرویس گوگل ادز است. در این روش شما یک وبلاگ یا سایت انگلیسی درست می کنید و با داشتن یک وبلاگ خوب کم کم مطرح می شین و از طریق تبلیغات گوگل به پولی قابل قبول (در رده دو تا سه میلیون در ماه) می‌رسین.

مشکل اینکار اما اینه که شما با «جهان» در رقابت هستین و منطقا باید حوزه ای رو انتخاب کنین که توش مطلب و مهارت کافی داشته باشین. در دنیای انگلیسی دزدیدن مطلب و کپی پیست سریعا وبلاگ شما رو با مشکلات متنوع مواجه می کنه و حتی ممکنه گوگل پرداخت به سایت شما رو متوقف کنه.

از اونطرف ما به عنوان یک ایرانی مشکل پرداخت و هویت داریم و باید مواظب باشین گوگل یکهو نفهمه شما از کشور عزیزمون هستین و در نهایت هم با بالا رفتن هزینه ها در ایران ما داریم کم کم عضو جهان می شیم و درآمد مثلا پونصد دلاری شما از سایتتون کم کم در ایران هم کم هیجان می شه. ولی کماکان این یکی از معقولترین کارهایی است که می تونین برای «کسب درآمد اینترنتی» انجام بدین.

کارهای واقعی از «طریق» اینترنت

خب یک شیوه «کسب درآمد از اینترنت» هم هست که عملا «از اینترنت» نیست و «از طریق» اینترنت است. مثلا من به عنوان یک برنامه نویس می تونم تو سایت هایی که فریلنسرها کار میکنن کار کنم. اونجا یک نفر می گه فلان برنامه رو لازم داره و من می گم با صد دلار براش می نویسم و قرارداد می بندیم و پولم رو می گیرم. یا مثلا یک جا به عنوان گرافیست کار می کنم و غیره و غیره. نمونه دیگه بورس و قمار و اینها است.

اونجا در اصل شما دارین به خاطر اینکه پوکر بازی خوبی هستین پول در میارین یا چون بورس باز خوبی هستین دارین بالا پایین رفتن پول رو پیش بینی میکنین و اینترنت فقط یک واسطه است. در همین طبقه بندی بذارین فروختن آثارتون رو توی اینترنت یا هر چیز دیگه. در این موارد هم باید حواستون باشه که شما باید یک وجه تمایز از آدم‌های دیگه داشته باشین که بتونین اینجا پول دربیارین. اینکه من فیلم گرگ وال استریت رو دیدم دلیل نمی شه که فکر کنم تو بورس حتما موفق خواهم بود.

بورس، پیش بینی، قمار، نویسندگی، ... همه و همه فن هستن و آدم ها توشون سال‌ها تمرین می کنن و چیز یاد می گیرن پس فقط چون من فکر می کنم خیلی باهوشم و تو درک اخبار خفنم، باعث نمی شه تو فارکس پولدار بشم. من شناخت خوبی از دوستانی که توی بورس‌های اینترنت کار می کنن دارم و اونهایی که خوب کار می کنن معمولا درآمدهایی معادل یک متخصص دارن که جایی استخدام شده و روزی هم تقریبا همونقدر کار می کنن.

سوال‌هایی که باید از خودتون بپرسین

قبل از شروع به یک بیزنس اینترنتی یا باور کردن نوشته‌های سایتی که به شما می گه می تونین فقط با خریدن یک کتاب یا یک مجموعه پولدار بشین باید از خودتون چند تا سوال بپرسین و اگر جوابشون رو داشتین قدم بعدی رو بردارین.

- آیا من مهارت خاصی دارم که بقیه ندارن؟ اگر می شه از این راه پولدار شد چرا اینهمه آدم در ایران و دنیا مشغول کارهای سخت تر هستن؟ فقط چون این سایت رو ندیدن؟
- نمونه های موفق این موضوع کجا هستن؟ درسته که نویسنده سایت عکس یک برج قشنگ در مالزی رو گذاشته و نوشته این خونه منه، اما آیا بقیه هم اینو تایید می کنن؟
- چرا می خوام اینکار رو شروع کنم؟ واقعا به نظرم خوبه یا فقط دنبال یک درآمد زیاد سریع راحت هستم و اینها خیال من رو با تکرار اینکه «اینجا راحت و سریع پول در می یاری» راحت کردن؟
- در نهایت به کجا می رسم؟ آیا ده سال بعد هنوز باید مشغول کپی پیست باشم که درآمد رویایی یک میلیون در ماهم رو حفظ کنم؟ یا چیزی یاد می گیرم که می تونم برم یک کار واقعی رو شروع کنم؟
- چرا طرف می خواد من رو هم توی روش پولدار شدنش شریک کنه؟ اگر واقعا پولدار شده و واقعا می خواد بقیه رو هم پولدار کنه دیگه چرا راه پولدار شدن رو مجانی نمی گه و هنوز لنگ این ده تومن من است و اینهمه تلاش میکنه من «محصول چگونه پولدار شویم» رو ده تومن و صد تومن ازش بخرم؟

تقریبا تمام ماجراهای «سریع و راحت پولدار بشین» معمولا یک الگوی مشترک دارن: یک نفر می گه یک کم پول به بده تا پولدارت کنم و بعد خودش با پولهایی که جمع کرده پولدار می شه. مثل همون نویسنده ای که کتاب «چگونه پولدار شویم» می نویسه تا مردمی که می خوان پولدار بشن، نفری ده هزار تومن بهش بدن که شاید پولدار بشه.

واقعیت اینه که سیستم‌های پولدار شدن اینترنتی تقریبا همشون مثل همون پرورش قارچ و ساخت عروسک توی خونه هستن. اینها می تونن پول بخور و نمیر یا فان برای یک دانش آموز در بیارن ولی اگر در خودتون قابلیتی می بینین بهتره دنبال یک کار، مهارت یا ایده واقعی بشین که در تمام عمر شما رو پیش ببره و بهش افتخار کنین.

یک نگاه به متخصص های واقعی بندازین و یک نگاه به کسانی که ادعای میلیاردر بودن از اینکارها دارن ولی هنوز شغل اصلی شون فروش سی دی فیلم و تبلیغ حلزون و کتاب راهنمای پولدار شدن است و ببینین دوست دارین شبیه کدوم گروه باشین.

۹.۵ راهنمای انتخاب زبان برنامه نویسی

جادی جان منظور من این بود چه نوع زبان برنامه نویسی از بین اون دوره ها خوبه که دوستم شرکت کنه؟ جاوا؟ اوراکل؟ دات نت؟ نمیخوام آموزشگاه بهم معرفی کنی میخوام یک زبان برنامه نویسی که میدونی الان بیشتر کاربرد و بازار کار داره رو بهم بگی.

ضرب‌المثلی در دنیای برنامه نویسی هست که می گه «اشتباهی که خیلی از برنامه‌نویس‌های تازه کار می کنن اینه که برنامه‌نویسی رو با یاد گرفتن کد نوشتن به یک زبان خاص اشتباه می‌گیرن». پس توصیه اول به این دوستمون اینه که دنبال یاد گرفتن برنامه نویسی باشه نه یاد گرفتن دستور زبان یک زبان خاص.

اکثر زبان‌ها در پایه به هم شبیه هستن و اگر یک زبون رو درست یاد بگیریم سوییچ کردن یا کد نوشتن به یک زبان هم‌خانواده چندان مشکل نیست. اینه که یاد گرفتن C می تونه پایه خوبی برای هر برنامه نویسی باشه و بعدش خوندن کد چند برنامه خوب (که روی گیت هاب به راحتی قابل دسترسی هستن و حتی می تونن به سادگی برنامه ای مثل `yes` در لینوکس باشن). نکته بعدی اینه که آدم‌ها به شیوه‌های مختلفی چیز یاد می گیرن. در کل به سه شیوه:

۱. خوندن (کتاب، راهنما، ...)

۲. آموزش دیدن (کلاس)

۳. انعکاسی (دیدن و تکرار کردن)

۴. تمرین کردن (پریدن وسط استخر و دست و پا زدن)

که البته مثل هر چیز دیگه‌ای، در دنیای واقعی هر کدوم با ترکیبی از روش‌های بالا به حداکثر یادگیری خودمون می‌رسیم. مثلا می دونم که بهترین روش یادگیری ام ترکیبی از خوندن و تمرین کردن است . این اصل ساده در آموزش گاهی به خاطر عادت ۱۲ ساله ما به مدرسه‌های کلاس محور ایرانی کلا فراموش می شه. پس قبل از ثبت نام کلاس یک لحظه باید به خودمون یادآوری کنیم که یاد گرفتن چیزها الزاما نیازمند کلاس نیست و اگر کل راهنماهامون رو از اینترنت بگیریم و کل هزینه کلاس رو خرج خوش گذرونی کنیم ممکنه خیلی بهتر چیز یاد بگیریم.

اما کدوم زبون؟ فرض کنیم دقیقا دنبال یاد گرفتن یک زبون برنامه نویسی هستیم. چه زبونی بهترینه؟ نقل می کنم از **سینا** ی عزیز که

تا وقتی اهداف رو ندونیم نمی تونیم مسیر رو ارزیابی کنیم.

هدف ما چیه؟ رسیدن به یک شغل مطمئن توی بازار کاری که از همیشه برامون کار هست؟ گرفتن پروژه‌های خاص با پول خیلی زیاد؟ مهاجرت؟ کار کردن توی یک استارتاپ مرتبط با گوشی‌های موبایل؟ اپلای کردن برای گوگل؟

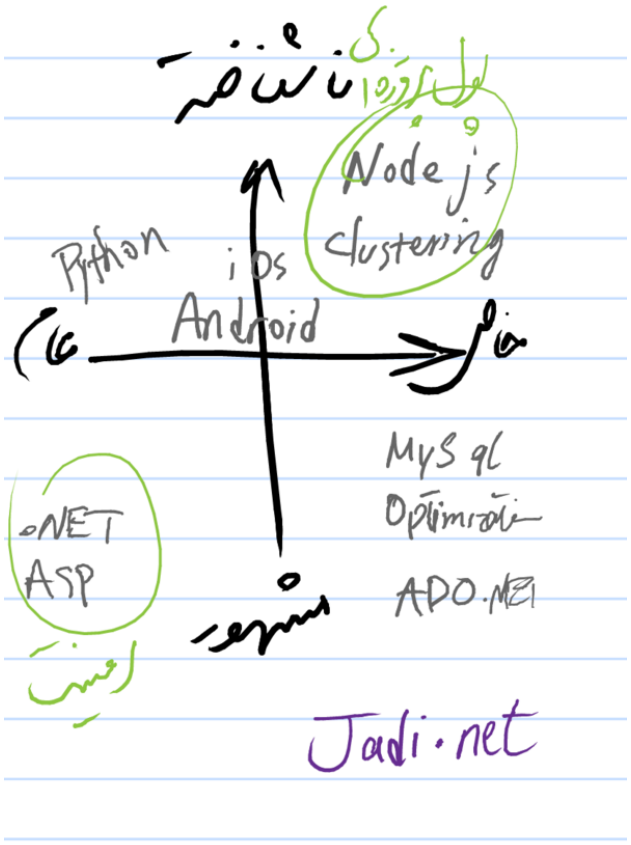
هدف رو اگر بدونیم تقریبا مشخصه که باید چیکار کنیم. راه بر اساس مقصد قابل تشخیصه و حتی در مواردی انتخاب مقصد به معنی انتخاب راه است.

بذار موضوع انتخاب بهینه زبان برنامه نویسی رو با یک نمودار متقاطع توضیح بدم. اطلاعات ما در جهان یا درباره حوزه‌های شناخته شده برای همه است که تبدیل می شیم به عضوی از یک خیل عظیم (مثلا بعد از چهار سال می شیم مهندس مخابرات فارغ التحصیل از دانشگاه) یا وقت گذاشتیم و در حوزه‌های کمتر شناخته شده برای عموم اطلاعات کسب کردیم (و مثلا شدیم کسی که تجربه و دانش زیادی در مورد چاپ کتاب داره). اولی شغل‌های بیشتری داره ولی معمولا حقوقش کمتره چون کلی آدم دیگه هم هستن که همونها رو بلدن.

از اونطرف یک بحث دیگه اینه که دانش ما چقدر عام است و چقدر خاص. آیا ما «کلا برنامه نویسی بلدیم» یا «می دونیم چطوری باید برنامه های چند رشته ای (مالتی ترد) نوشت» ؟ آیا ما کلا کتاب چاپ کردن بلدیم یا تخصصمون دقیقا در این است که بدونیم بهترین چسب موجود در بازار برای چسبوندن جلد کتاب به شیرازه، چیه. بذارین این دو تا رو روی نموداری که حرفش رو می زدم نشون بدم: با توضیات بالا واضحه که یک برنامه نویس دات نت همیشه حقوق داره و همیشه در شرکت های متوسط کار داره. استرسش برای پیدا کردن کار کمه ولی در عوض در نگه داشتن کار اوضاعش خوب نیست چون هزاران نفر هر سال مجموعه مدرک های MCTS MCSE و غیره رو می گیرن و می شن برنامه نویس دات نت. در مقابل به بخشی نگاه کنین که با «پول پروژه‌ای» مشخص شده. ما الان در شرکت دو ماهه دنبال کسی می گردیم که به شکل پروژه‌ای بیاد برای ما سرورهای نود جی اس رو کلاستر و High Aavailable کنه و کم اهمیت ترین موضوع در پروسه قرارداد اینه که طرف چقدر پول می خواد. احتمالا طرف با این کار چند روزه می تونه به اندازه چند ماه برنامه نویس دات نت پول در بیاره ولی ظاهرا در کشور عزیز افراد خیلی خیلی کمی هستن که این کار رو بلد باشن چون هم در حوزه ناشناخته است و هم در حوزه تخصصی.

حالا فکر می کنم انتخاب براتون راحت تر باشه. من همیشه در حوزه ناشناخته تخصصی بودم و راستش یک محور دیگه هم در جدول کشیدم: ترکیبش با یک چیز دیگه. مثلا متخصص سیستم عاملی که مخابرات بلده. مدیر پروژه‌ای که لینوکس بلده و ... این خیلی کم پیدا می شه و خیلی هم مورد نیاز نیست ولی اگر کسی شما رو بخواد... واقعا شما رو همه جوره می خواد.

در عوض ممکنه شما به این نتیجه برسین که یک کار امن و راحت می خواین و در این صورت باید نگاهی به بخش استخدام شرکت ها بندازین و ببینین الان چی بورسه. ممکنه دات نت باشه و ممکنه جاوا باشه یا اگر تصمیمتون این شده که برین سراغ استارتاپ موبایلی شخصی خودتون، معلومه که باید یا iOS یاد بگیرین یا Android Development که یک جاهایی بین جدول بالا افتاده.



توجه: جدول رو قارتی کشیدم. اصلا معنی اش این نیست که کاملا اندیشیده و دقیق است. می شه در مورد همه اجزاش حرف زد.

جایزه اینکه تا اینجا خوندین : اگر هدفتون رفتن به چیزی مثل گوگل است، برین و آگهی‌های شغلی اش رو ببینین و بعد دقیقا می دونین که در طول یکی دو سال آینده باید چی یاد بگیرین و به کدوم پروژه‌ها کمک کنین تا خود گوگل بیاد ازتون خواهش کنه که برین تو شرکتش کار کنین (: این دیدن ترند مهمه. لازمه درک کنیم که در جهان آینده چه چیزهایی دارن رو می یان و براشون آماده باشیم[۱] تا اینجا خوندین؟[۲].

پی نوشت آخر . گفتیم که «اشتباه برنامه نویس های جوان اینه که برنامه نویسی رو با برنامه نویسی به یک زبون خاص اشتباه میگیرن». این یعنی اگر شما برنامه نویسی رو یاد بگیرین می تونین بعدا به هر چیز دیگه سوییچ کنین. رو پایه ها تمرکز کنین و تا وقتی درک می کنین که دارین برنامه نویسی یاد میگیرین،

[۱] مرتبط طنز:آیا موفقیت زبان های برنامه نویسی به ریش و سبیل سازندگانش مربوطه؟

[۲] (: پس شاید دوست داشته باشین این شماره ویژه پادکست در مورد برنامه نویسی رو هم گوش بدین.

نگران زبون نباشین.

و البته به فهرست محبوبیت زبان‌های برنامه نویسی در جهان هم نگاهی بندازین و اگر هنوز سوالی هست در کامنت ها مطرح کنین تا سعی کنم از برنامه نویس هایی بسیار خوب بخوام جواب هاش رو براتون بگن.

صادق توی کامنت‌ها نوشته: ضمن تایید حرف‌های جادی می‌خواستم برای کسایی که تازه می‌خوان شروع کنند یه چند تا نکته را یادآوری و تاکید کنم:

۱. سعی کنید با زبانی مثل `python` شروع کنید که هم زود بتونین نتیجه بگیرین و هم عادت‌های خوب کدنویسی براتون نهادینه بشه.

۲. از خوندن و یادگرفتن `Design pattern`ها و `Best practice`ها ولو با زبان دیگه‌ای غیر از زبان تخصصی شما پیاده‌سازی شده غفلت نکنید که موجب پشیمانیست.

۳. از ابتدای کار عادت کنید برای کدهاتون کامنت مناسب و واضح بنویسید.

۴. کد خوب بخونید (توی پروژه‌های آزاد اغلب کدها خیلی خوبند چون تعداد زیادی توش مشارکت می‌کنند)، توی توسعه پروژه‌های آزاد مشارکت کنید از گزارش باگ گرفته تا نوشتن پلاگین و ...

۵. سعی کنید ارتباط‌هاتون با آدم‌های متخصص توی زمینه کاریتون زیاد کنید، از فرصت گپ زدن باعاشون استفاده کنید، توی فروم و `irc` فعال باشین نه فقط برای پرسیدن که برای مشارکت و پاسخ دادن.

۶. خوشبین باشید و شاد. کاری را انجام بدین که ازش لذت می‌برین. البته لذت بردن خیلی وقت‌ها به معنی آسون بودن یا پول زیاد درآوردن نیست. در ضمن سعی نکنید تقلید کنید.

۷. وقتی دارین کد می‌خونید واقعا بفهمید چی به چیه. و بد نیست فکر کنید روش بهتری برای پیاده‌سازی اون کد به ذهنتون می‌رسه یا نه.

۸. وقتی تونستید کد بقیه را سریع دیباگ کنید می‌تونید کم‌کم حس کنید که حرفه ای شدین. البته یادگرفتن انتها نداره.

برای بحث بیشتر به اینجا مراجعه کنین[۱]

[۱] `jadi.net`

۱۰.۵ بیانیه هکرها

مانیفست هکر یا بیانیه هکر یا The Hacker Manifesto متنی است که هکری به نام منتور (Mentor) در سال ۱۹۸۶ نوشته است. این بیانیه کوتاه پس از دستگیری منتور توسط پلیس نوشته شده و در نشریه زیرزمینی فرک شماره ۱، نسخه ۱ فایل ۳ از ۱۰ چاپ شد. این بیانیه یکی از متون مرجع فرهنگ هکرها است و خواندن آن می‌تواند نمایانگر روحیه و دیدگاه هکرهای نسل اول به دنیا باشد. مانیفست هکرها، راهنمایی است برای هکرهای تمام دنیا و زیربنای کوتاهی برای اخلاقیات پذیرفته شده در این جامعه که می‌گوید توانایی‌های فنی به جای مقاصد خودخواهانه و صدمه زننده به دیگران، باید برای ساخت و گسترش مرزهای جهان آزاد بکار برده شود.

در زیر متن بیانیه هکرها را به آن شکلی که در مجله فرک ۸ ژانویه ۱۹۸۶ منتشر شده می‌خوانید:

وجدان یک هکر

نوشته +++منتور+++

نوشته شده در ۸ ژانویه ۱۹۸۶ امروز یک نفر دیگر دستگیر شد. همه روزنامه‌ها در مورد آن نوشته‌اند. «نوجوانی در رسوایی جرایم کامپیوتری دستگیر شد»، «هکر بعد از دستکاری در بانک دستگیر شد»، ... .

بچه‌های لعنتی. همه مثل هم هستند.

اما آیا شما هرگز در برنامه‌های روانشانسی و تکنومغزهای ۱۹۵۰تان به عمق چشم‌های یک هکر نگاه کرده‌اید؟ آیا هیچ وقت فکر کرده‌اید که چه چیزی او را ساخته و چه افکاری به او شکل داده؟

من یک هکر هستم، وارد دنیای من شوید...

دنیای من با مدرسه شروع شد... من از اکثر دانش‌آموزها باهوش‌تر بودم و مزخرفاتی که درس می‌دادند حوصله من را سر می‌برد...

لعنت به احمق‌ها. همه مثل هم هستند.

من در دبیرستان هستم. به معلم گوش می‌دهم که برای دفعه پانزدهم مشغول توضیح روش‌های کاهش اصطکاک است. من این را می‌فهمم. «نه خانم اسمیت. من نمی‌توانم مشقم را روی کاغذ نشان بدهم. من آن را در ذهنم حل کرده‌ام...»

بچه لعنتی. احتمالا آن را کپی کرده. آن‌ها همه مثل هم هستند.

من امروز یک کشف کردم. یک کامپیوتر کشف کردم. یک لحظه صبر کنید! این عالی است. هر کاری که به آن بگویم می‌کند. اگر من اشتباهی کنم دلیلش این است که من گند زده‌ام و نه به این خاطر که من را دوست ندارد...

یا به این خاطر که از من می‌ترسد...

یا به این خاطر که فکر می‌کند من باهوشم...

یا به این خاطر که درس دادن را دوست ندارد و نمی‌خواهد اینجا باشد...

بچه لعنتی. تنها کاری که می‌کند بازی کردن است. همه مثل هم هستند.

و بعد اتفاق افتاد... دری به جهانی باز شد... پالسی الکتریکی مانند ماده ای اعتیاد آور از خط تلفن خارج شد و من را از ناتوانی‌های رومزه‌ای که می‌دیدم خلاص کرد.

«همین است... این جهانی است که من به آن تعلق دارم...» من اینجا همه را می‌شناسم... حتی اگر آن‌ها را ندیده باشم، حتی اگر هیچ وقت با آن‌ها حرف نزده باشم، شاید در آینده هم هیچ وقت خبری از آن‌ها نگیرم اما همه آن‌ها را می‌شناسم...

بچه‌های لعنتی. دوباره گند زده‌اند به خط تلفن... همه‌شان مثل هم هستند.

شما فکر می‌کنید می‌دانید که همه مثل هم هستیم... در روزهایی که ما هوس استیک داشتیم، در مدرسه با قاشق به ما غذای بچه می‌دادند. تکه گوشت‌هایی که به ما می‌دادید قبلا جویده شده بودند و مزه‌ای نداشتند. ما توسط سادیست‌ها احاطه شده بودیم و آدم‌های مریض به ما بی‌توجهی می‌کردند. بعضی‌ها هم بودند که چیزهای خوبی برای درس دادن داشتند اما آن‌ها قطره‌هایی بودند نایاب در بیابانی بی‌انتها.

حالا این دنیای ما است... دنیای الکترون و سوییچ و زیبایی پهنای باند. ما از سیستم‌های موجود بدون اینکه پول بدهیم استفاده می‌کنیم ولی اگر به خاطر آن شکم‌پرست‌های سودجو نبود که لازم داشته باشند پولشان را صرف این کنند که رسانه‌ها به ما مجرم بگویند، این سرویس‌ها باید بسیار ارزانتر بودند. ما کشف می‌کنیم... و شما به ما مجرم می‌گویید. ما به دنبال دانش می‌گردیم... و شما به ما مجرم می‌گویید. ما بدون رنگ پوست، بدون ملیت و بدون گرایشات مذهبی در دنیا زندگی می‌کنیم و شما به ما مجرم می‌گویید. شما بمب اتم می‌سازید، شما جنگ شروع می‌کنید، شما می‌کشید، شما تقلب می‌کنید و به ما دروغ می‌گویید و سعی می‌کنید ما باور کنیم که این چیزها برای ما خوب است، اما ما هستیم که مجرمیم.

بعله من مجرمم. جرم من کنجکاوی است. جرم من قضاوت کردن در مورد انسان‌ها نه بر اساس ظاهر که بر اساس آنچه می‌گویند و آنچه فکر می‌کنند. جرم من این است که از شما باهوش‌ترم، جرمی که هرگز به خاطر آن مرا نخواهید بخشید.

من یک هکرم، و این بیانیه من است. شما شاید بتوانید این یک نفر را متوقف کنید اما نمی‌توانید جلوی همه ما را بگیرید... به هرحال، همه ما مثل هم هستیم.

+++ منتور +++

۱۱.۵ چگونه هکر شویم

این مقاله‌ای بسیار مهم است از هکر بزرگ **اریک ریموند** با ترجمه **آرش بیژن‌زاده** و کمی اصلاحات برای انتشار مجدد

هکر (hacker) کیست؟

پرونده اصطلاحات[۱] شامل تعاریفی از "هکر" است که عموما در ارتباط با تعریف تکنیکی آن همراه با وصف سرخوشی‌حل مشکلات و مرتفع کردن محدودیت هاست. ا

گر شما می‌خواهید بدانید چگونه هکر شوید تنها دو تعریف به دردتان می‌خورد:

یک اجتماع ، یک فرهنگ مشترک ، از برنامه نویسان خبره و جادوگران شبکه که پیشینه آن از میان دهه ها به مینی‌کامپیوترهای‌اشتراک زمانی[۲] و اولین تجربه های ARPAnet می‌رسد. اعضای این فرهنگ اصطلاح "هکر" را ساختند . هکرها اینترنت را ایجاد کردند. آنان سیستم عامل Unix را آنچنان که امروز هست ایجاد کردند. هکرها usenet را اجرا کردند. آنان باعث شدند شبکه جهانی (World wide web) کار کند. اگر شما دارای این فرهنگ هستید و دیگران می‌دانند که شما چه کسی‌هستید و هکر می‌نامندتان؛ پس شما هکر هستید!

نگرش هکر محدود به "هکرنرم‌افزار" نمی‌شود ، مردمانی‌هستند که منش هکر را به سایر زمینه ها مانند الکترونیک یا موسیقی‌سرایت می‌دهند. در حقیقت شما می‌توانید این فرهنگ را در بالاترین سطوح هر علم و هنری بیابید. هکرهای نرم‌افزار این هم روحان را در هر جا می‌شناسند، آنان را "هکر " می‌نامند و برخی‌معتقدند که طبیعت هکر واقعا مستقل از زمینه ایست که او در آن کار می‌کند.

اما در تمام این مقاله ما بطور خاص بر مهارت ها و منش های‌هکر نرم‌افزار، و فرهنگ های مشترکی که واژه ی "هکر" را بوجود آورد تمرکز خواهیم کرد. گروه دیگری‌از مردم هستند که متکبرانه خود را هکر می‌نامند اما نیستند! این مردمان (که بیشتر نرهای نابالغند) کسانی‌هستند که سیستمهای‌کامپیوتری و مخابراتی را "تخریب" می‌کنند.

هکرهای واقعی‌اینان را "شکننده"[۳] می‌نامند و هیچ کاری‌به آنان ندارند. هکرهای واقعی اعتقاد دارند که اینان تنبل، بی‌مسئولیت و نه چندان باهوشند و می‌دانند که توانایی نفوذ به سیستمهای امنیتی‌شما را هکر نمی‌کند. همانگونه که دزدان اتومبیل را هیچگاه نمی‌توان مکانیک نامید. متأسفانه بسیاری از روزنامه نگاران و نویسندگان ناآگاهانه واژه ی" هکر" را برای‌توصیف شکننده ها (Crackers) بکار می‌برند و هکرها را تا

---

[۱] Jargon File
[۲] time-sharing
[۳] Cracker

سرحد مرگ عصبانی می‌کنند.

تفاوت اصلی این است: هکرها می‌سازند اما شکننده‌ها ویران میکنند.

اگر می‌خواهید هکر باشید (همواره) مطالعه کنید. اما اگر می‌خواهید شکننده شوید گروه خبری `alt.2600` را بخوانید و آماده باشید که ۵ تا ۱۰ سال را در زندان بگذرانید، پس از اینکه فهمیدید به اندازه‌ای که فکر می‌کردید زرنگ نیستید. این تمام چیزی است که درباره ی‌شکننده ها (`Crackers`) خواهم گفت.

منش هکر

هکر می‌آفریند و یاری‌می‌کند. او به آزادی و یاری متقابل معتقد است؛ برای آن که هکر نامیده شوید باید چنان رفتار کنید که گویا چنین منشی‌دارید و برای‌اینکه اینگونه رفتار کنید باید واقعا آن را داشته باشید. اگر به پروراندن منش هکر تنها برای پذیرفته شدن در این فرهنگ می‌اندیشید در اشتباه هستید! چنین منشی داشتن همواره کمکتان میکند یادگیرید و با انگیزه باشید. مانند تمام هنرها بهترین راه استاد شدن، نگاه کردن به استاد و تقلید از اوست - نه فقط در باب تفکر که حتی‌در احساس!

همانگونه که در شعر ذن زیر آمده است:

> تا که راه یابی
> به استاد نگر
> به دنبالش باش
> با او برو
> از نگاه او بنگر
> استاد شو

پس اگر میخواهید استاد شوید، آن قدر ذکرهای‌زیر را بگویید (افکار زیر را با خود مرور کنید) تا باورشان کنید:

۱ - جهان پر از مشکلات جذابی‌است که میباید حل گردند.

هکر بودن هیجان دارد، اما هیجانی که نیازمند تلاش فراوان است و تلاش کردن نیازمند انگیزه. ورزشکاران موفق انگیزه خود را از لذتی که در جسمشان احساس میکنند، می‌گیرند؛ در گذر از حدود جسمانیشان. شما نیز باید از حل مشکلاتتان مشعوف شوید. از پیشرفت مهارتتان و زورآزمایی اندیشه‌ی‌تان. اگر شما به طور ذاتی‌چنین شخصی‌نیستید باید این‌گونه گردید و گرنه انرژیتان با شهوت، پول، شهرت و ... به هدر خواهید داد.

همچنین باید به توانایی یادگیریتان ایمان آورید - باور به اینکه: گر چه تمام آن چه را لازم دارید نمی‌دانید اما اگر تنها بخشی از آنرا کشف کنید توانایی حل باقی را بدست می‌آورید

۲ - هیچ مشکلی‌نباید دوبار حل گردد اندیشه های‌خلاق گرانبها و محدودند.

ذهن های خلاق باارزش هستند، منابعی محدود. آنان نباید با دوباره کشف کردن چرخ، به هدر بروند در حالی که هزاران معمای حل نشده جذاب باقی است. برای آن که کرداری مانند یک هکر داشته باشید باید باور کنید که وقت هکرها گرانبهاست - آنچنان که گویی یک وظیفه روحانی است که اطلاعتان را مبادله کنید، مشکلات را حل کنید و راه حل‌ها را به دیگر هکرها بدهید تا آنان مشکلات جدیدتر را مرتفع کنند. بجای آنکه دائما حول همان‌ها بگردند.

مجبور نیستید باور داشته باشید که تمام آنچه را خلق می‌کنید باید ببخشید. گرچه هکرهایی که چنین می‌کنند محترم‌ترین آنان هستند - با ارزش های هکر سازگار است که مقداری از آن را بفروشید تا برای خود خورد و خوراک و کامپیوتر تهیه کنید. چه خوب است اگر استعداد هکری خود را برای حمایت از خانواده و حتی‌ثروتمند شدن بکار گیرید، تا هنگامی‌که شرافت هنرتان و رفیقان هکرتان را فراموش نکنید.

۳- کسالت و بیکاری شیاطینند.

هکرها (و عموما انسانهای خلاق) هرگز نباید کسل شوند یا مجبور به بیگاری‌شوند چرا که در این صورت آنها از انجام کاری که تنها آنان قادر به انجام آنند باز می‌مانند. این هرز رفتن همه را آزار می‌دهد. بنابراین کسالت و بیگاری نه تنها ناخوشایند بلکه واقعا شیاطینند.

برای اینکه مانند یک هکر رفتار کنید باید باور کنید که می‌خواهید تمام کسالت آوران را کنار بزنید نه تنها برای خودتان بلکه برای همه (خاصه سایر هکرها).

یک استثناء بارز وجود دارد. هکرها گاهی کارهایی انجام می‌دهند که به نظر تکراری و خسته کننده می‌رسند، تنها برای پاکسازی ذهن شان یا برای بدست آوردن تجربه‌ای خاص. اما به اختیار ،هیچ فرد اندیشمندی نباید مجبور به پذیرفتن موقعیت کسل کننده‌ای‌گردد.

۴ - آزادی‌خوب است.

هکرها ذاتا ضد استبدادند. هر که به شمادستور دهد، شما را از پرداختن به آنچه عاشق کشف آنید باز می‌دارد؛ گر چه آنان همواره برای دستوراتشان دلایل ابلهانه‌ی خود را دارند. با منش استبدادی باید مبارزه شود هر جا که پیدا شود چرا که شما و تمام هکرها را تحت فشار می‌گذارد.

این به مفهوم مخالفت کلی‌با اتورتیه نیست. کودکان باید راهنمایی شوند و جنایتکاران مراقبت. هکر ممکن است نوعی از اتورتیه را قبول کند تا بیشتر از زمانی که برای‌اجرای دستورات از دست می‌دهد، بدست آورد. اما این تنها یک معادله آگاهانه است. یک قدرت فردی که مستبدان می‌خواهند قابل قبول نیست.

مستبدان تنها با سانسور و پنهان کاری رشد می‌کنند و به همکاری داوطلبانه اعتمادی ندارند. آنها فقط نوعی از همکاری را دوست دارند که در کنترل آنان باشد، بنابراین برای اینکه یک هکر باشید باید دشمنی ذاتی

با سانسور و رازداری، اعمال قدرت و سیاست برای مجبور ساختن افراد متعهد را در خود بپرورانید و باید که براین باور عمل کنید.

۵ - منش جایگزین مهارت نیست.

برای آنکه هکر شوید مجبور به پروراندن مواردی از این منش‌هایید. اما تنها تقلید از این منشها شما را هکر نمی‌سازد همانگونه که شما را قهرمان یا خواننده نمی‌کند. هکر شدن احتیاج به تیزهوشی، تمرین، تمرکز و کار سخت دارد.

بنابراین شما باید یاد بگیرید که به منش مشکوک و به مهارت احترام بگذارید. هکرها نمی‌گذارند فضولان وقت شان را هدر دهند و به مهارت ایمان دارند به خصوص مهارت در درک کردن گرچه مهارت در هر زمینه ای‌دلپذیر است. مهارت در زمینه های مورد نیاز که متخصصان کمتری دارد بهتر است و تخصص در زمینه های مورد نیاز که به فکر متبحر، استادی و تمرکز نیاز دارند، بهترین.

اگر شیفته مهارت باشید از پروراندن آن در خود لذت خواهید برد . کار سخت و تمرکز بجای‌کاری‌کسالت بار به بازی‌سخت شبیه می‌گردد و این برای‌هکر شدن حیاتی‌است .

مهارت‌های پایه‌ای برای هکر شدن

منش هکر حیاتی‌است اما مهارت او حیاتی‌تر است. منش جایگزین مهارت نمی‌گردد و مجموعه مهارت های پایه ای خاصی وجود دارند که باید در خود بپرورانید تا هکرها شما را هکر بنامند.

این ابزار به آرامی تغییر میکند با گذشت زمان تکنولوژی مهارت های جدیدی ایجادمی‌کند و قدیمی‌ها را بی‌مصرف میکند. مثلا در گذشته زبان ماشین شامل این مجموعه بود، در حالیکه HTML اخیرا به این مجموعه اضافه شده است. اما این مجموعه در حال حاضر مشخصاً شامل موارد زیر است:

۱ - بیاموزید که چگونه برنامه بنویسید

این مسلما پایه ای‌ترین مهارت هکر است. اگر شما هیچ زبان برنامه نویسی‌بلد نیستید، پیشنهاد می‌کنم با پیتون شروع کنید. پتیون تمیز طراحی‌شده است. به خوبی‌مستند سازی‌شده و تقریباً ابتدایی‌است. با آنکه پتیون زبان اولیه خوبی‌است، یک اسباب بازی‌نیست. بلکه بسیار قدرتمند و قابل انعطاف است و مناسب پروژه های بزرگ. من یک مقاله مفصل تر در مورد انقلاب پیتون نوشته‌ام. خودآموزهای خوبی می‌توانید در وب سایت پیتون پیدا کنید.

جاوا زبان بسیار خوبی‌است البته بسیار مشکل تر از پیتون است، اما برنامه سریع تری ایجاد می‌کند و فکر می‌کنم انتخاب فوق العاده ای‌برای‌زبان دوم است .

البته شما نمی‌توانید با دانستن تنها دو زبان به مهارتهای یک هکر و یا حتی یک برنامه نوبس خوب برسید. شما باید بدانید چگونه در مورد مشکلات برنامه نویسی، جدای از زبان برنامه نویسی‌فکر کنید. برای آنکه یک هکر واقعی شوید، باید به جایی برسید که بتوانید یک زبان جدید را در دو، سه روز یاد بگیرید. این نیازمند آن است که چندین زبان کاملاً متفاوت را یاد بگیرید.

اگر به صورت جدی به برنامه نویسی روی آورید، باید C زبان پایه‌ای Unix را یاد بگیرید. C++ بسیار شبیه C است؛ اگر شما یکی از آنها را یاد بگیرید، یادگرفتن دیگری مشکل نخواهد بود. اما هیچ کدام به عنوان زبان اول قابل یادگیری نیستند. در واقع، هر چقدر از برنامه نویسی به زبان C پرهیز کنید، بازده‌تان بیشتر خواهد بود.

C بسیار کاراست و منابع کامپیوتر را کمتر مصرف می‌کند. متأسفانه C این کارایی را با تلاش بسیار شما برای مدیریت سطح پائین منابع (مانند حافظه) بدست می‌آورد. این نوع برنامه نویسی سطح پائین بسیار پیچیده و باگ -دوست است و زمان بسیاری برای رفع اشکال[۱] لازم دارد. با قدرت و سرعتی که کامپیوترهای امروز دارند این معامله خوبی نیست. تیز هوشانه‌تر است که از زبانی استفاده کنیم که زمان کامپیوتر را بیشتر می‌گیرد و زمان برنامه نویس را کمتر . مانند ، پیتون. زبانهای دیگری هستند که ارزش خاصی برای هکرها دارند.

Perl[۲]: به جهت دلایل کاربردی آن با ارزش است، به طور گسترده ای در طراحی صفحات فعال وب و مدیریت سیستم به کار گرفته شده است و حتی اگر شما هرگز با Perl برنامه ننوبسید، باید قادر به خواندن آن باشید. بسیاری از مردم از Perl در جایی استفاده می‌کنند که من پیتون را پیشنهاد کردم. برای اجتناب از برنامه نویسی C در جاهایی که نیاز به کارایی C ندارید. شما به فهمیدن کدهای Perl احتیاج خواهید داشت.

LISP[۳]: به دلایل دیگری ارزشمند است . برای روشن نگری عینی که پس از یادگیری آن بدست خواهید آورد. حتی اگر هیچگاه از لیسپ به طور جدی استفاده نکنید، مسلما یادگیری آن شما را برنامه نویس بهتری خواهد کرد. (شما می‌توانید مهارت های اولیه ی LISP را به راحتی بانوشتن و تغییر دادن Mod ها برای ویرایشگر متن Emacs کسب کنید).

حقیقتا بهتر است هر پنج زبان (پیتون ، جاوا ، C/C++ ، پرل و LISP) را یاد بگیرید. جدا از ارزشی که این زبانها برای هکرها دارند، آنان رویکردهای کاملا متفاوتی برای برنامه نویسی دارند که مسا ئل با ارزشی به شما یاد می‌دهند.

نمی‌توانم دستورالعمل خاصی برای یادگرفتن برنامه نویسی بدهم (کار پیچیده ای‌است)، اما می‌توانم بگویم که کتاب‌ها و کلاسها به شما کمک نخواهند کرد (اکثر هکرها خودشان یاد گرفته‌اند) شما می‌توانید روشهایی را از کتاب فراگیرید اما ساختار فکری که این روشها را به مهارت واقعی تبدیل می‌کند، تنها با تمرین و شاگردی

---

[۱] Debug
[۲] Perl.com
[۳] lisp.org

کردن بدست می‌آید. وظایف شما شامل:

۱. خواندن کد

۲. و نوشتن کد

خواهد بود.

یادگیری زبان برنامه نویسی مانند یادگیری نوشتن یک زبان واقعی است. بهترین راه خواندن چیزهایی است که استادان امر نوشته اند و سپس نوشتن برنامه ای از خودتان است؛ بسیار بیشتر مطالعه کنید، کمی‌بیشتر بنویسید، بیشتر مطالعه کنید، بیشتر بنویسید... و تا آنجا ادامه دهید که نوشته هایتان قدرت و صلابت کارهای استادتان را پیدا کند.

سابقا پیدا کردن کد خوب مشکل بود، برنامه های بزرگی که متن آنها در دسترس بود تا هکرها بخوانند و آزمایش کنند، بسیار محدود بود. اکنون این مسئله به طور قابل ملاحظه‌ای تغییر کرده است؛ اکنون نرم‌افزارها ابزارهای‌برنامه نوبسی و سیستمهای‌عامل بازمتن (که تماما بوسیله هکرها نوشته شده است) بسادگی قابل دسترس است - که مرا به نوشتن بخش بعد ترغیب می‌کند...

۲- یکی‌از یونیکس های باز - متن را بگیرید و استفاده و اجرا کردن آن را بیاموزید

فرض می‌کنیم یک کامپیوتر شخصی دارید یا لااقل به آن دسترسی دارید[۱]. مهمترین قدم اولیه ای که هر مبتدی برای هکر شدن می‌تواند بردارد، گرفتن یک کپی از لینوکس (Linux) یا بی‌اس دی-یونیکس (BSD-Unix)؛ نصب کردن آن روی‌کامپیوتر شخصی و اجرای آن است.

بله ، سیستم عاملهای فراوانی در کنار یونیکس وجود دارد. اما تمام آنها به صورت باینری توزیع می‌شوند و شما قادر به خواندن و تغییر کد آن نیستید. تلاش برای ایجاد تغییر بر روی یک کامپیوتر داس یا ویندوز یا MacOS مانند این است که بخواهید در لباس شوالیه رقص بیاموزید.

تحت OS/X این کار ممکن است، اما فقط بخشی از این سیستم بازمتن است. شما به موانع بسیاری برخورد خواهید کرد و باید مواظب باشید تا عادت بد تکیه بر کد اختصاصی اپل را در خود توسعه ندهید. در صورتی که بر روی یونیکس‌ها تمرکز کنید، چیزهای مفیدتری فراخواهید گرفت.

یونیکس سیستم عامل اینترنت است. اگر اینترنت را بدون یونیکس یاد می‌گیرید، هیچ وقت نمی‌توانید یک هکر اینترنت باشید. به این خاطر فرهنگ هکر امروز کاملاً یونیکس-محور است. (این مطلب همیشه صادق نبوده است . بسیاری از هکرهای با سابقه از این موضوع رضایت ندارند اما پیوند محکم یونیکس و اینترنت آنچنان قوی است که حتی قدرت شیطانی مایکروسافت نیز نتوانسته است خلل چندانی در آن ایجادکند.)

---

[۱] بچه های امروزی خیلی راحت به آن دسترسی دارند:-)

پس یک یونیکس نصب کنید. من به شخصه لینوکس را دوست دارم اما راههای دیگری هم وجود دارد (بله! شما می‌توانید مایکروسافت ویندوز و Linux رابا هم داشته باشید).

یاد بگیرید، اجرا کنید، ور بروید، کدهایش را بخوانید و تغییرشان دهید. ابزار برنامه نویسی بهتری در اختیار خواهید داشت، مانند C، LISP، Python و Perl که در سیستم عامل ویندوز خواب داشتن آن ها را می‌بینند. بسیار جذاب و سرگرم کننده خواهد بود و آنچنان در دانش غرق میشوید که حتی متوجه آن نمی‌شوید تا هنگامیکه به مانند یک استاد هکر به پشت سرتان بنگرید!

برای اطلاعات بیشتر درباره ی یادگیری Unix به The Loginataka[۱] نگاه کنید. همین طور شمامی‌توانید نگاهی به The Art Of Unix Programing[۲] (هنر برنامه نویسی در یونیکس) بیاندازید. برای آن که چیز هایی از لینوکس دست گیرتان شود به سایت Linux Online[۳] بروید؛ شما می‌توانید از آن جا دانلود کنید یا (ایده ی‌بهتر) یک گروه کاربران لینوکس محلی پیدا کنید تا به شما در نصب لینوکس کمک کنند. از دیدگاه یک کاربر تازه کار تمام توزیع های لینوکس بسیار شبیه یکدیگرند. شما می‌توانید راهنما و منابع BSD Unix را در سایت bsd.org[۴] پیدا کنید. من نیز مقالاتی مبتدی درباره ی پایه های‌یونیکس و لینوکس[۵] نوشته ام.

> توجه : من در حقیقت نصب کردن هیچ کدام از Linux یا BSD ها را به طور خاص به شما توصیه کنم ، برای هر تازه کاری هر کدام از این ها یک پروژه ی انفرادی است. برای لینوکس، یک گروه کاربران لینوکس در محل خود پیدا کنید و از آنها برای کمک سوال کنید.

۳ - استفاده از وب و نوشتن HTML را یاد بگیرید

بسیاری از چیزهایی که فرهنگ هک ساخته است خارج از افق دید شماست، کمک به کارخانه‌ها، دفاتر و دانشگاه‌ها بدون اینکه تأثیر مشخصی در زندگی غیر هکرها نداشته باشد. در این میان اینترنت یک استثناء عمده است، سرگرمی درخشان هکری که حتی به اعتراف سیاست مداران در حال تغییر دادن جهان است. تنها به همین خاطر (و همچنین بسیاری‌از دلایل مشابه دیگر) یاد گرفتن کار در اینترنت احتیاج دارید.

این فقط به این معنی نیست که چگونه از یک مرورگر استفاده کنید(!) بلکه به معنی یادگیری HTML است . اگرهنوز برنامه نویسی یاد نگرفته‌اید، نوشتن HTML عادت های ذهنی را برایتان فراهم می‌کند که به یادگیری برنامه نویسی کمک می‌کند. پس برای خودتان یک Homepage درست کنید. سعی کنید

---

[۱] http://catb.org/~esr/faqs/loginataka
[۲] http://catb.org/~esr/writings/taoup
[۳] www.linux.org
[۴] www.bsd.org
[۵] Unix and Internet Fundamentals

از XHTML استفاده کنید که نسبت به HTML سنتی تمیزتر است.(منابع بسیار خوبی برروی وب برای تازه کارها وجوددارد [۱])اما نوشتن یک Homepage به هرحال آنقدر خوب نیست که شما را هکر کند. وب پر از Homepage است. بیشترشان بی‌ارزشند. لجن های بی‌محتوا، فضولات شیک، اما مطمئن باشید که لجن همیشه لجن است. (برای اطلاعات بیشتر صفحه ی TheHTML Hell[۲] را ببینید.) برای با ارزش بودن؛ Homepage تان باید محتوا داشته باشد و برای هکرهای دیگر جذاب و یا آموزنده باشد. تمام اینها شمارا به بخش بعد هدایت می‌کند...

۴ - اگر انگلیسی بلد نیستید آن را یاد بگیرید

به عنوان یک آمریکایی بخاطر آنکه زبان مادریم انگلیسی است قبلاً از ذکر این موضوع ناراحت بودم. حداقل این می‌تواند یک امپریالیسم فرهنگی‌تلقی گردد. ولی تعدادی از غیر انگلیسی زبانان از من خواستند که این موضوع را متذکر شوم که انگلیسی زبان فرهنگ هکر و اینترنت محسوب می‌گردد و شما احتیاج خواهید داشت که این زبان را یادبگیرید تا در جامعه هکرها فعال شوید.

این موضوع واقعیت دارد. حدود سال ۱۹۹۱ متوجه شدم که بسیاری از هکرها که انگلیسی‌زبان دومشان بود آن را برای بحث های تکنیکی‌شان بهره می‌گرفتند، حتی اگر زبان مادریشان یکی بود. به من اطلاع دادند که انگلیسی بعلت غنی‌تر بودن به لحاظ لغات فنی برای این کار مناسب‌تر است . به همین دلیل ترجمه متن‌های فنی که در زبان انگلیسی هستند، غالبا رضایت بخش نیست. لینوس توروالدز که یک فنلاندی است، کد خود را به زبان انگلیسی تشریح کرده است (و هرگز غیر از این روش، روش دیگری را پیش نگرفته است) تسلط بر انگلیسی، عامل مهمی در جمع کردن جامعه جهانی برنامه نویسان لینوکس بوده است. این مورد نمونه قابل ذکری در مورد نقش زبان انگلیسی است.

موقعیت فرهنگ هکر

مانند بسیاری از فرهنگ‌های برپایه ی روابط غیر اقتصادی، هکرگری نیز با شهرت اداره می گردد، شما سعی می‌کنید که مسئله جالبی را حل کنید اما اینکه آن مسئله چقدر قابل تأمل است یا راه حل شما واقعا چقدر خوب است، چیزی است که تنها استادان شما صلاحیت تأیید آن را دارند.

به همین ترتیب، وقتی وارد بازی هکر ها شدید، مدارجتان را با آنچه سایرین در مورد شما فکر می کنند بدست خواهید آورد (به این علت است که تا هنگامی که دیگران شما را هکر نمی دانند واقعاً هکر نیستید). این حقیقت بوسیله پنداری که هک را یک کار منزوی گرایانه می‌داند، محو شده است؛ هم چنین با وجود این تابوی فرهنگ هکری (که در حال از میان رفتن ولی فعلا همچنان نیرومند است) در برابر پذیرش اینکه تصدیق

---

[۱] این یکی از آن هاست make-a-web-site.com!

[۲] catb.org/~esr/html-hell.html

خود یا دیگری، تنها در گیر انگیزه یک شخص باشد.

به خصوص ، هکرگری نوعی از فرهنگ است که مردم شناسان به آن فرهنگ هدیه می‌گویند. شما شهرت و موقعیت خود را نه با سلطه بر دیگر مردم، نه با زیبایی یا در اختیار داشتن چیزهای مورد نیاز مردم بلکه با دادن هدیه بدست می آورید. به خصوص با دادن وقت خود، خلاقیت و مهارتتان.

پنج چیز وجود دارد که با انجام آن مورد احترام هکرها قرار می‌گیرید:

۱ - برنامه‌های باز - متن بنویسد

اولین (محوری ترین و سنتی ترین) روش، نوشتن برنامه‌هایی است که هکرهای دیگر آن را جالب و مفید می‌دانند و سپس دادن کد منبع برنامه‌ها به دیگران. (ما قبلا این را **نرم‌افزار آزاد** می‌نامیدیم ، اما این اصطلاح موجب اشتباه بسیاری از مردم شد که نمی‌دانستند منظور از آزاد دقیقا چیست، امروزه بسیاری از ما حداقل به نسبت ۲ به ۱ اصطلاح بازمتن[۱](open-source) را ترجیح می دهیم. محترم ترین هکر ها[۲] افرادی هستند که برنامه‌های بزرگی نوشته اند - برنامه‌های پرقدرتی که احتیاجات گسترده ای را مرتفع می سازد - و آنان را در دسترس همگان قرار داده اند.

۲ - به آزمایش ورفع اشکال کردن برنامه‌های بازمتن کمک کنید

هکرها به کسانی که نرم افزارهای بازمتن را آزمایش و رفع اشکال می‌کنند، یاری میرسانند. در این دنیای ناقص ناگزیر به صرف دقت بسیاری برای رفع اشکال برنامه‌ها هستم، به این علت است که مولفان بازمتن میگویند یک آزمایشگر خوب (تعریف کردنش دشوار است؛ مشکلات در ضمن انتشار، کسیانی که بتواند اشتباهات یک انتشار عجله ای را تحمل کنند و مشکلات نرم‌افزار را گزارش کنند) سزاوار یاقوت به اندازه ی وزنشان هستند.

حتی یک نفر از آنان میتواند رفع اشکال کردن را از یک کابوس طولانی به یک دردسر عبرت آموز تبدیل کند. اگر مبتدی هستید یک نرم افزار در حال برنامه نویسی پیدا کنید و یک آزمایشگر خوب باشید. یک پیشرفت طبیعی از کمک به آزمایش برنامه تا کمک به رفع اشکال کردن آن و بهتر کردن آن است. از این راه چیزهای بسیاری یاد می‌گیرید و روابط خوبی با افرادی که بعدا شما را کمک خواهند کرد بر قرار خواهید کرد.

۳ - اطلاعات خوب را منتشر کنید

کار خوب دیگری که می‌توانید بکنید جمع آوری و دستچین کردن مطالب جالب و مفید در برگه‌های وب یا پرونده‌هایی مانند سؤالات متداول (FAQ) و منتشر کردن آن است. گردآورندگان مجموعه سؤالات متداول (FAQ) به اندازه برنامه نویسندگان بازمتن مورد احترام هستند.

---

[۱] opensource.org

[۲] یاد داشت مترجم : در نوشته ی اریک ریموند این واژه نیم -خدا (demi-god) نوشته شده بود.

۴ - به پایداری شالوده ی کار کمک کنید

فرهنگ هکر (و مهندسی اینترنت بعنوان شاخه‌ای از آن) با داوطلبان به پیش می رود. بسیاری از کارهای کوچک ولی ضروری وجود دارند که باید انجام شوند. مدیریت لیست‌های پستی و گروههای خبری، مرتب کردن آرشیو نرم افزارهای بزرگ، گسترش RFC ها و سایر استانداردهای فنی. مردمی که این کارها را انجام می دهند مورد احترام فراوان هستند. چرا که همه می دانند این نوع مسئولیت چقدر زمانبر است در حالیکه جذابیت زیادی مانند بازی کردن با کد هم ندارد. انجام آنها نشاندهنده ی ایثارگریست.

۵ - به خود فرهنگ هکر کمک کنید

در انتها می توانید به خود فرهنگ کمک کنید و آن را منتشر کنید[۱]. گرچه این کاری نیست که در همان ابتدا انجام دهید تا وقتی که شهرت خوبی در بین هکرها بدست آورید.

فرهنگ هکر، رهبر به معنی دقیق آن ندارد. اما قهرمانان، پیران ، مورخان و سخنگویان زیادی دارد. بعد از این که به اندازه ی کافی در سنگرها مدت زیادی را سپری کنید، می‌توانید یکی از آنها شوید. باید بدانید که هکرها به منیت آشکار پیران خود بدبینند؛ رسیدن به این درجه از شهرت آشکارا خطرناک است. به جای تلاش برای رسیدن به آن موقعیتتان را چنان بسازید که در مسیرشما افتد. سپس در مقامتان فروتن و مهربان باشید.

رابطه هکر / نرد (Nerd)

ارتباط هکر و نرد بر خلاف افسانه مشهور، برای هکر بودن اجباری بر نرد بودن نیست [۲]. اما به هرحال نرد بودن کمکتان می‌کند و بسیاری از هکرها اینگونه‌اند. نرد بودن کمکتان می‌کند که بر مهمترین مسائل مانند فکر کردن و هک کردن تمرکز داشته باشید.

به همین خاطر بسیاری از هکرها صفت نرد بودن و حتی سرسختانه‌تر گیک را به عنوان شعار برگزیده‌اند. روشی برای بیان جداییشان از انتظارات عوامانه اجتماع - برای بحث بیشتر به صفحه‌ی geek[۳] مراجعه کنید. اگر شما بتوانید به اندازه کافی روی هک کردن تمرکز کنید در حالیکه به زندگیتان هم برسید، بسیار عالیست. امروزه انجام این کار از ۱۹۷۰ که من تازه کار بودم بسیار ساده تر است؛ جریان غالب فرهنگی با تکنو- نردها بسیار مهربانتر است و تعداد کسانی که می فهمند هکرها عاشقان و همسران بلند مرتبه‌ای هستند هر روز زیادتر می شود.

اگر شما بخاطر نرد بودنتان به هکر بودن علاقه‌مند شده‌اید هم خوب است! حداقل برای متمرکز شدن مشکلی نخواهید داشت. شاید هم در آینده از انزوا درآمدی!

---

[۱] مثلاً با نوشتن مقاله ای در مورد اینکه چگونه هکر شویم :-)

[۲] نرد به شخصی گفته می‌شود که تمام زندگی او بر کامپیوتر/تکنولوژی استوار است -مترجم -

[۳] samsara.circus.com

نکاتی در باب طریقت

نکاتی در باب طریقت باز می‌گویم که شما برای هکر شدن باید ساختار فکری هکری بدست آورید. چیزهایی هست که هنگامیکه کامپیوتر ندارید می‌توانید انجام دهید. آنها جایگزین هک کردن نمی شوند (هیچ چیز نمی‌شود) اما بسیاری از هکرها انجامشان را دوست دارند و احساس می کنند با انجام آنها به نوعی به روح هک کردن نزدیک می شوند.

بیاموزید که زبان مادریتان را خوب بنویسید. گرچه معروف است که برنامه نویس ها نمی‌توانند بنویسند، یک تعداد غافلگیر کننده‌ای از هکرها (تمام بهترین هکرهایی که من می‌شناسم) نویسندگان توانایی هستند.

- داستانهای علمی - تخیلی بخوانید. به جلسات داستانهای علمی بروید. (جای خوبی که می توانید هکرها و هکر دوستان را ببینید.)
- ذن تمرین کنید و/ یا به هنرهای رزمی بپردازید (انظباط روحی در جهات بسیاری شبیه‌اند)
- گوش تان را به موسیقی حساس کنید. بیاموزید که به نوع خاصی از موسیقی را درک کنید. نواختن برخی آلات موسیقی را به خوبی فرابگیرید یا آواز خواندن یاد بگیرید.
- کار با جملات قصار و بازی با کلمات را به خوبی بیاموزید.

هر چه موارد بیشتری را قبلاً انجام داده باشید استعداد بیشتری برای هکر شدن دارید. چرا این موارد خاص مهم هستند واقعا معلوم نیست. ولی ارتباط آنها با مهارتهای نیمکره ی چپ و راست مغز مربوط می شود، هردوی این ها اهمیت فراوانی دارند؛ هکرها همانگونه که به منطق استدلالی نیاز دارند به شهود عرفانی نیز محتاجند تا در لحظه ای خاص از شر منطق ناقص مشکلی خلاص شوند.

به همان میزان که بازی می‌کنید، کارکنید و همان قدر که کار می‌کنید، بازی کنید. برای هکرهای واقعی مرزی میان " بازی" ، " فعالیت" ، "دانش" و "هنر" وجود ندارد و این با پدیدار شدن سطح بالایی از سرزندگی سازنده همراه خواهد بود. به هیچ وجه به اطلاعات مهارت‌های محدود اکتفا نکنید. برخلاف آن که بسیاری از هکرها خود را یک برنامه نویس معرفی می کنند، دارای مهارت‌های بسیاری هستند - مدیریت سیستم، طراحی وب و رفع اشکال‌های سخت‌افزاری PC یکی از معمول ترین آن هاست. هکری که مدیر سیستم است، اغلب، یک برنامه نویس حرفه‌ای و یک طراح وب است. هکر هرگز کاری را نیمه انجام شده رها نمی‌کند، اگر به موضوعی بپردازد در رابطه با این موضوع مهارت‌هایش را به اوج کمال می‌رساند.

در پایان چیزهایی هستند که نباید انجام دهید:

- از اسامی ابلهانه و بزرگ نما (قلمبه!) استفاده نکنید.
- در آتش افروزیهای گروه‌های خبری و یا هر بحث بی فایده ی دیگر شرکت نکنید.

- خودتان را "ولگرد سایبر" خطاب نکنید، وقت خود را با چنین افرادی هدر نکنید.
- نامه‌های الکترونیکی پر از غلط املایی و دستور زبانی نفرستید.

تنها چیزی که از این ها عایدتان می شود شهرت یک دلقک است. هکرها حافظه خوبی دارند - سالها طول می کشد تا دسته گلی که به آب داده‌اید فراموش شود و مورد قبول واقع شوید. بر مشکل نام‌های کاربری یا اسامی مستعار باید تاکید کنم. پنهان کردن نام واقعی پشت رموز، کار ابلهانه و بچه گانه کرکر ها(crackers) و warez d00dz و یا دیگر فرم‌های پیش پا افتاده ی زندگیست. اگر نام مستعاری دارید آن را دور بیاندازید. در میان هکرها این حقیقتا باعث می‌شود تا شما را به فراموشی بسپارند. هکران از آنچه که انجام می‌دهند مغرورند و آن را وابسته به نام حقیقی خود می‌خواهند.

منابع دیگر

پیتر سیباج[۱] برای مدیران سیستمی که نمی دانند چطور با هکرها سر کنند، یک FAQ مکمل نوشته است که [۲] Hacker FAQ نام دارد. اگر سایت Peter پاسخ دهی نمی‌کند، جستوجوی سایت Excite[۳] می‌تواند یک کپی خوب برای شما پیدا کند.

در سایت mines.edu یک سند وجود دارد[۴] که How To Be A Programmer (چگونه یک برنامه نویس شویم) نام دارد، این یکی از بهترین و کامل ترین هاست. ارزش این مستند فقط مربوط به آموزش کد نویسی نیست، در این سند درباره ی کد نویسی به صورت گروهی و چالش‌های یک کد نویسی گروهی صحبت شده است.

من مقاله ای به نام تاریخ اجمالی هکرگری[۵] نیز نوشته‌ام.

برای آشنایی با فرهنگ لینوکس و بازمتن مقاله‌ای با نام "کلیسای فقید و بازار"[۶] یا The Cate-dral and the Bazaar نوشته‌ام. ادامه ی این مقاله در مقاله ای به نام Homesteading Noosphere[۷] آمده است.

Rick Moen، مقاله‌ای به نام "یک گروه کاربران لینوکس چطور به کار می‌افتد؟"[۸] نوشته است.

---

[۱] Peter Seebach
[۲] plethora.net/~seebs/faqs/hacker
[۳] search.excite.com
[۴] HowToBeAProgrammer.pdf
[۵] A Brief History Of Hackerdom
[۶] Cathedral-Bazaar
[۷] Homesteading Noosphere
[۸] How to Run A Linux User Group

باز هم از Rick Moen و من (اریک ریموند) مقاله‌ای به نام "چگونه یک سوال هوشمندانه بپرسیم؟"[۱]، وجود دارد.

اگر شما به اطلاعات پیش‌نیاز برای کامپیوترهای شخصی و شبکه ی اینترنت احتیاج دارید، به مقاله ی "پایه‌های یونیکس و اینترنت"[۲] مراجعه کنید.

اگر شما برنامه‌ای منتشر می‌کنید و یا وصله‌ای برای برنامه‌ای می نویسید، به "راهنمای تمرین انتشار برنامه‌ها" [۳] سر بزنید.

اگر شما به اشعار ذن علاقه مند هستید، احتمالا باید از"The Unix Koans of Master Foo"[۴] خوشتان بیاید.

سؤالاتی که زیاد پرسیده شده اند

سوال: به من یاد می دهید چطور هک کنم ؟

جواب: از اولین روز انتشار این برگ هر هفته (گاهی هر روز) چندین درخواست از مردم بدستم می‌رسد که: همه چیز هک کردن را به من یاد بدهید! متأسفانه وقت و انرژی کافی برای این کار ندارم. پروژه‌های هکری من و مسافرتهایم بعنوان مدافع بازمتن روزی ۱۱۰٪ وقتم را می‌گیرد. حتی اگر هم می‌توانستم؛ هک کردن هنر و منشی است که شما خود باید یاد بگیرید. بعدا متوجه خواهید شد که با آنکه هکرها دوست دارند به شما کمک کنند، اما اگر بخواهید همه چیز را حاضر و آماده در دهان شما بگذارند، تحویلتان نمی‌گیرند.

اول خودتان چیزهایی یاد بگیرید. نشان دهید که دارید سعی می کنید، که توانایی یاد گرفتن دارید سپس به سراغ هکرها بروید و پرسشهایتان را مطرح کنید.

اگر می‌خواهید به هکری نامه ی الکترونیکی بفرستید باید از قبل دو چیز را بدانید. اولین چیز این که ما متوجه شدیم که کسانی که در نوشته‌هایشان بی دقت‌اند معمولا تنبل‌تر از آنند که هکرهای خوبی بشوند. بنابراین مواظب غلط‌های املایی و انشایی خودتان باشید و گرنه شما را نادیده می‌گیرند. دوم این که هرگز جواب نامه‌ی الکترونیکی خود را در آدرسی غیر از آدرسی که از آن نامه می‌فرستید نخواهید. ما می‌دانیم که کسانی که این کار را می کنند دزدانی اند که از حساب دزدی استفاده می کنند و هیچ علاقه ای به کمک کردن به دزدها نداریم.

سوال: خوب پس از کجا شروع کنم ؟

---

[۱] How to Ask Smart Questions
[۲] The Unix and Internet Fundamentals HOWTO
[۳] Software Release Practice HOWTO
[۴] Rooties Root: The Unix Koans of Master Foo

جواب: بهترین راه برای شروع رفتن به جلسه یک لاگ(گروه کاربران لینوکس[۱]) است[۲].

به احتمال قوی می‌توانید یکی از آنها را در حوالی خود بیابید که احتمالاً وابسته به یک دانشگاه یا مؤسسه است. اعضای لاگ احتمالاً به شما یک نسخه از لینوکس می‌دهند و حتماً کمکتان می‌کنند که آنرا نصب کنید.

سوال: کی باید شروع کنم ؟ آیا خیلی دیر نشده است ؟

جواب: در هر سنی که علاقه‌مند شدید می‌توانید یاد بگیرید. اکثر مردم در سن ۱۵ تا ۲۰ سالگی علاقه‌مند می شوند؛ من استثناهایی را از هر دو طرف می‌شناسم.

سوال: چقدر طول می کشد تا هکر شوم ؟

جواب: بستگی به این دارد که چقدر باهوشید و چقدر پشت کار دارید. اگر مصمم باشید معمولاً ظرف ۱۸ تا ۲۴ ماه می‌توانید مهارتی قابل ملاحظه بدست آورید. اما کار به اینجا ختم نمی‌شود. اگر یک هکر واقعی هستید تمام عمرتان را صرف یادگیری و تکمیل هنرتان خواهید کرد.

سوال: آیا `Visual Basic` و `C#` (سی شارپ) زبان‌های خوبی برای شروع کارند ؟

جواب: اگر شما این پرسش را مطرح می‌کنید یعنی به هک کردن تحت سیستم عامل مایکروسافت ویندوز فکر می‌کنید. به خودی خود تفکر بدی است. یاد گیری کد نویسی تحت پلتفورم ویندوز مانند یاد گیری رقص است وقتی زره به تن کرده‌اید، من چندان خوشم نمی‌آید. به آن جا نروید. آن بخش بی‌نهایت کثیف است و از کثافت باید پرهیز کرد.

زبان‌های `Visual Basic` و `C#` مشکلات مخصوص خودشان را دارند؛ در اصل این‌ها غیر قابل انتقال یا `not portable` هستند. هیچ نسخه‌ی بازمتنی از این زبان‌ها وجود ندارد. استانداردهای اجرایی `ECMA` چیزی بیش از تعدادی رابط برنامه نویسی را پوشش نمی‌دهند. در ویندوز بیشتر کتابخانه‌ها از یک سازنده ی تنها (مایکروسافت) که مالک آن است پشتیبانی می‌کنند؛ اگر شما بی‌نهایت نسبت به چیزی که استفاده می کنید دقیق نباشید، برای همیشه به پلتفورم مایکروسافت وابسته خواهید شد. برای این که به این باطلاق فرو نروید، کسی که کد نویسی را شروع می کند باید بسیار دقت کند. اگر شما روی یونیکس شروع کنید زبان‌های بهتر و کتابخانه‌های بهتر وجود دارند.

بنابراین به همان سیستم عامل که سازنده زبان انتخاب می کند، میخکوب می‌شوید، این به مذاق هکرها سازگار نیست.

به خصوص `Visual Basic` بسیار مضر است. مانند همه بیسیک‌ها `Visual Basic` هم بسیار بد طراحی شده است، نه از من نخواهید که این موضوع را تشریح کنم؛ این موضوع می‌تواند یک کتاب را پوشش دهد. یک زبان را که به خوبی طراحی شده است فرا بگیرید.

یکی از عادت‌های برنامه نویسی بدی که به شما یاد خواهد آموخت وابستگی به کتابخانه ها، `widget`

---

[۱] `Linux user group LUG`

[۲] این گروهها را می توانید درسایت `LDP` بیابید.

ها و ابزار برنامه نویسی یک شرکت خاص است. به طور کل هر زبانی که تحت لینوکس یا یکی از نسخه‌های BSD پشتیبانی نگردد و یا حداقل توسط سه شرکت مختلف پشتیبانی نگردد، ارزش این را ندارد که برای هک کردن یاد بگیریدش.

سوال: به من یاد می دهید که چطور یک سیستم را بشکنم؟

جواب: نه. کسی که بعد از خواندن این مقاله هنوز این سؤال را بپرسد، احمق تر از آن است که یاد بگیرد، حتی اگر من وقتش را داشته باشم. هر میلی که چنین درخواستی کند نادیده گرفته می شود یا با خشونت تمام پاسخ داده می‌شود.

سوال: چطور می توانم رمز عبور شخص دیگری را بدست آورم؟

جواب: این شکستن است (crack). گم شو احمق!

سوال: چطور می توانم پست الکترونیکی شخصی دیگری را بخوانم / واردش شوم / تحت نظر بگیرمش؟

جواب: این شکستن است . سریع گم شو ...

سوال: چطور می توان کانالهای chat را دزدید؟

جواب: این شکستن است . مردک احمق!

سوال: سیستمم را شکسته‌اند ! کمک می‌کنید از خودم دفاع کنم؟

جواب: نه! هر بار که از من این سؤال را پرسیده‌اند از طرف یک کاربر بیچاره ی ویندوز بوده است. امکان ندارد ویندوز را به طور کامل امن کنید. کد و معماری آن پر از ایراد است و سعی در ایمن کردن آن آب در هاون کوبیدن است. تنها راه پیش‌گیری، رفتن به یک سیستم عامل دیگر مانند لینوکس یا حداقل سیستم عاملی است که توانایی ایمن شدن را داشته باشد.

سوال: من با ویندوزم مشکل دارم . کمکم می کنید؟

جواب: البته! به خط فرمان بروید و بنویسید:

```
format c:
```

ظرف چند دقیقه تمام مشکلاتتان حل می‌شود.

سوال: کجا می توانم با هکرهای واقعی صحبت کنم؟

جواب: بهترین جا لاگ‌های محلی خودتان است. لیست شان را می توانید در سایت LDP[1] بیابید. (قبلاً در می‌گفتم در IRC هیچ هکر واقعی پیدا نمی‌کنید. ولی وضعیت فرق کرده است. به طور مشخص اجتماعاتی از هکرهای واقعی مربوط به GIMP و Perls کانالهای IRC وجود دارند)

سوال: چند کتاب مفید در زمینه هکر کردن پیشنهاد کنید؟

---

[1] tldp.org

جواب: لیست خواندنی‌های لینوکس ممکن است مفید باشد. Loginntaka هم می‌تواند جالب باشد. برای پیش در آمدی بر پیتون در Python.org نگاهی بیاندازید.

سوال: آیا باید در ریاضیات خوب باشم تا بتوانم هکر شوم ؟

جواب: نه . البته باید قادر باشید به طور منطقی فکر کنید و رشته درست دلایل را دنبال کنید، هک کردن نیاز بسیار اندکی به ریاضیات رسمی دانشگاهی دارد. به خصوص، شما معمولا به آنالیز و جبر نیاز نخواهید داشت (این را به مهندسان برق واگذار کنید). داشتن پیش زمینه‌ای در رشته‌هایی از ریاضیات مانند منطق، تئوری مجموعه‌ها ، نظریه اعداد و.... ممکن است مفید باشد.

آنچه بسیار مهم است : شما باید بتوانید منطقی تفکر کنید و رشته ای از دلایل صحیح را گرد هم بیاورید، کاری که ریاضی‌دانان انجام می‌دهند. زمانی که ارتباط با اغلب علوم ریاضی کمکی به شما نکرد، شما به نظم و ذکاوت برای پیش برد آن نیاز خواهید داشت. اگر شما چندان باهوش (با ذکاوت) نی‌ستید، امید چندانی به هکر شدن شما نیست؛ اگر شما نظم فکری نداشته باشید این امید به همان میزان کم خواهد شد.

یک راه خوب برای فهمیدن این موضوع گرفتن و مطالعه یک کپی از کتاب Raymond Smullyan است که اسم آن اکنون یادم نیست. مشابه چیستان‌های منطقی و بامزه Smullyan در روح هکرگری خیلی وجود دارد. توانایی در حل آنها علامت خوبی است و لذت در حل کردن آن علامتی بهتر.

سوال: چه زبانی را باید اول یاد بگیرم ؟

جواب: XHTML (آخرین نسخه ی HTML) اگر هنوز آنرا بلد نیستید. کتابهای بد زیادی هستند با توضیحات خسته کننده بلندبالا و چند کتاب خوب وجود دارد[1].

البته HTML یک زبان کامل برنامه نویسی نیست. وقتی برای یادگیری آماده شدید، پیشنهاد می‌کنم از پیتون شروع کنید. خیلی‌ها به شما Perl را پیشنهاد می‌کنند و هنوز محبوبیت بیشتری دارد ولی یاد گرفتن آن سخت است و (به نظر من) به خوبی پیتون نیست.

C واقعا مهم است ولی از Perl و پیتون خیلی سخت‌تر است. سعی نکنید اول C را یاد بگیرید. کاربران ویندوز، به دام VB نیفتند. عادتهای برنامه نویسی بدی به شما یاد می‌دهد و قابل انتقال به هیچ سیستم دیگری غیر از ویندوز نیست.

بپرهیزید!

سوال: چه سخت افزاری نیاز دارم ؟

جواب: قبلا کامپیوترهای شخصی سرعت و حافظه کمی داشتند که همین‌ها کافی بودند تا روند یاد گیری هکر رامحدود کنند. این مساله مدتهاست که از بین رفته است ، هر کامپیوتری از Intel 486DX50 بالاتر باشد برای برنامه نویسی کافیست، X ، و ارتباطات اینترنتی ، و کوچکترین دیسک سخت به اندازه کافی بزرگ است.

---

[1] چیزی که من بیشتر از همه دوست دارم HTML: The Definitive Guide

مهمترین مساله‌ای که در انتخاب سخت افزار وجود دارد این است که آیا با لینوکس همخوانی دارد؟ (یا با BSD همخوانی دارد). البته برای بسیاری از کامپیوترهای جدید این همخوانی وجود دارد مگر در مورد تعدادی از مودم‌ها و چاپگرها که مخصوص ویندوز طراحی شده اند.

یک FAQ درباره ی سازگاری سخت افزار ها وجوددارد؛ آخرین نسخه ی آن اینجاست.

سوال: باید از مایکرو سافت متنفر باشم ؟

جواب: نه! نه اینکه میکروسافت نفرت‌انگیز نیست؛ مساله این است که فرهنگ هکر مدتها قبل از مایکروسافت وجود داشته است و مدتها بعد از آن هم خواهد بود. انرژیی را که برای نفرت از مایکروسافت صرف می‌کنید، برای عشق به هنرتان مصرف کنید. اگر برنامه‌ای خوب بنویسید، مشت محکمی است بر دهان مایکروسافت و خونتان را را کثیف نمی‌کند.

سوال: ولی بازمتن برنامه نویس‌ها را بیکار نمی‌کند؟

جواب: حقیقتا برعکس این است، تا بحال صنعت بازمتن بیشتر اشتغال زایی کرده است تا از بین بردن آن. اگر داشتن یک برنامه آماده اقتصادی تر از نداشتن آن است، در هر حال یک برنامه نویس حقوقش را می گیرد چه برنامه بازمتن باشد یا نباشد و مهم نیست چقدر نرم افزار آزاد هست ، به نظر می رسد همیشه تقاضای بیشتری برای نرم افزارهای جدید یا تطبیق داده شده وجود دارد. من در این باره در ورق‌های باز متن [۱] بیشتر خواهم نوشت.

سوال: چطور شروع کنم ؟ از کجا یک یونیکس آزاد پیدا کنم ؟

جواب: قبلا راجع به پیدا کردن یونیکس‌های معمول آزاد توضیح داده ام. برای هکر شدن به انگیزه خلاقیت و قابلیت خود آموزی نیاز دارید. پس دیگر شروع کنید...

اریک ریموند نسخه ی اصلاحی (Reversion) شماره ی ۱.۲۹

منبع[۲]: catb.org

ترجمه : آرش بیژن‌زاده

ویرایش و اصلاحات: نوید عبدی، آلن باغومیان

انتشار اول: www.technotux.com

بازنشر: کیبرد آزاد - جادی دات نت [۳]

---

[۱] Open Source

[۲] www.catb.org

[۳] برای بحث بیشتر به این مجموعه کامنت در jadi.net مراجعه کنین

۱۲.۵ چگونه فلان چیز رو یاد بگیرم

«چطوری پایتون یاد بگیرم؟»، «چطوری توسعه کرنل رو یاد بگیرم؟»، «چطوری هک کنم؟» منظورم چگونه هکر شویم [۱] اریک ریموند نیست، منظورم کشف حفره‌های امنیتی در کرنل است»، «چطوری دانشمند داده بشم؟» و ... این‌ها سوال‌هایی هستن که همیشه تکرار می‌شن و واقعا هم جواب ثابتی ندارن، یا بهتره بگم جوابشون بنا به شرایط هر فرد تغییر می کنه.

به زودی به این «حالت های مختلف یادگیری» نگاه خواهیم کرد ولی قبلش لازمه به یک نکته اشاره کنم: خیلی وقت ها سوال اصلی این نیست که «چگونه X رو یاد بگیرم» بلکه اینه که «من می خوام X باشم». وقتی کسی می پرسه «چطوری می تونم کرنل رو توسعه بدم» واقعا منظورش این نیست که «من می خوام چند سال برنامه نویسی یاد بگیرم و ساعت‌های طولانی در روز لیست پستی کرنل رو بخونم و سوادم رو بالا ببرم و یک بخش حوصله سر بر ولی مفید (مثلا مهندسی معکوس پروتکل یک کارت شبکه بی سیم غیرمرسوم و پیاده سازی اون به زبان سی) رو انجام بدم و بعد سعی کنم یک نفر دیگه رو قانع کنم که این بخش رو قبول کنه و به توروالدز پیشنهاد بده» بلکه منظورش اینه که «من می خوام خفن باشم، راه ساده ای داره؟». همین مساله در مورد دنیای هک هم هست.

یک هکر کلاه سفید به شکل طبیعی کسی است که ساعت ها و روزها و ماه‌ها وقت می ذاره و می گه فلان لوپ در فلان روتین خاص شاید فلان مشکل رو داشته باشه یا کلی وقت می ذاره یک قرارداد تست نفوذ می بنده و بعد از روی یک لیست بلند بالا یکی یکی حالت های مختلف نفوذ رو روی یک نرم افزار تست می کنه و گزارشی تایپ می کنه که توش می گه چه مشکلات امنیتی در این نرم افزار هست و طبق قرارداد پولش رو می گیره یا شغلش اینه که کلی فرمور کلی روتر رو آپدیت کنه و این چیزها ولی خیلی ها دنبال **تصویری غیر واقعی** هستن که توش نصفه شب ها بیدارن و کار می کنن (و خوابشون هم نمی یاد و خسته هم نیستن) و می تونن به ایمیل همه مردم دنیا دسترسی داشته باشن و هر حساب بانکی که خواستن رو هک کنن و غافلن از اینکه اصولا در دنیا چنین آدمی وجود نداره.

خلاصه اینکه نکته اول اینه که بدونیم اصولا آیا سوال اینه که «چطور فلان چیز رو یاد بگیرم» یا سوال اینه که «می خوام فلان چیز باشم، راه ساده ای هست؟».

همیشه نقل می کنم از استاد پیران که می گفت آسانسور پیشرفت هنوز اختراع نشده و هر کس می خواد پیشرفت کنه باید قدم به قدم از پله‌های ترقی بالا بره.

اما اگر سوال واقعا اینه که «چطور می تونم فلان چیز رو یاد بگیرم؟» جواب اصلی اینه: به چهار طریق.

---

[۱] linuxbook.ir

چهار شیوه یادگیری

در **علم یادگیری** نظریات بسیار متنوعی است که من معمولا به یکی از سر راست ترین هاش ارجاع می دم؛ نظریه پیتر هانی و آلن مامفورد. این دو نفر در نوشته‌هاشون می‌گن که آدم‌ها در چهار مرحله / روش یاد می‌گیرن:

۱. فعال

۲. انعکاسی

۳. تئوری

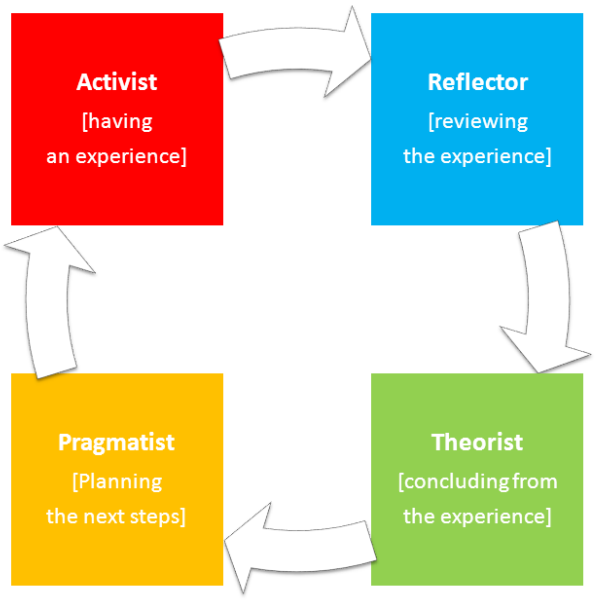
۴. عملگرایانه

این چهار روش می تونن در یک چرخه با هم ترکیب بشن. بنا به این نظریه بعضی افراد با نگاه کردن به رفتار دیگه یاد می گیرن (خوندن کد یا نگاه کردن به برنامه نویسی دیگران)، بعضی‌ها با یادگیری تئوری و بعد اجرا (مثلا خوندن یک کتاب پایتون)، بعضی ها با روش فعال (یادگیری و اجرای چیزهایی که یاد گرفتن) و در نهایت گروهی با روش عملگرا (پریدن توی استخر! شروع کردن به برنامه نویسی و یادگرفتن هر چیزی که لازمه). من خودم رو بیشتر جزو گروه آخر می دونم. در سیستم من اگر قراره چیزی رو یاد بگیرم اول یک مساله براش پیدا می کنم (مثلا قراره بشمرم که در هر قوطی کبریت به طور متوسط چند تا کبریت هست) و بعد شروع به کار می کنم و اجزای لازم برای کارم رو حین کار سرچ می کنم و می خونم و یاد می گیرم. مثلا حتی ممکنه در شروع به کار با یک زبان جدید مثل جولیا لازم باشه سرچ کنم «`for loop in julia`» و برنامه رو پیش ببرم.

نکته مهم اینه که شما کشف کنین روش یادگیری شما چیه و بر اساس اون پیش برین. مثلا کسی که به یکی از شیوه‌های انعکاسی (دیدن کار دیگران و تئوریزه کردن اون) یا تئوری یاد می گیره ممکنه کلاس براش بهترین روش ممکن باشه و در مقابل برای من که روشم عملگرایانه است، تقریبا کلاس بی معناترین چیز ممکن است و می تونم قسم بخورم که تا حالا هیچ چیز فنی رو از کلاسی یاد نگرفته ام.

در ضمن لازمه شما بعد از شناخت خودتون، به روش های دیگه هم دقت کنین. مثلا اگر من تونستم با یادگرفتن چند کتابخونه و سرچ برنامه ام رو بنویسم، باید حواسم باشه که حتما وقت کافی برای مطالعه یک کتاب در مورد پایتون یا آر یا جولیا یا هر چیزی که دارم یاد می گیرم هم بخونم تا سوادم منحصر نشه به چیزهایی که در مسیرم بوده یا پیش اومده یا اگر کسی از طریق تئوری یاد می گیره لازمه به کدهای بقیه هم نگاه کنه و خودش هم برنامه بنویسه تا چرخه یادگیری رو کامل کرده باشه.

پس اگر سوال واقعا این است که «چطوری ایکس رو یاد بگیریم» جواب من اینه که اول کشف کنین که چجور روش یادگیری برای شما مفیدتره، بعد با حوصله و قدم به قدم توش پیش برین. یادتون باشه که تقریبا



هر چیزی نیاز به تلاش داره و عملا هیچ چیزی نیست که بدون گذروندن چندین روز و بعد چندین ماه بشه توش حرفه ای بود. اگر در زمانی خیلی سریع دارین یاد میگرین بدونین که احتمالا دارین اشتباه می کنین یا اصولا اون چیز ارزش یاد گیری نداره چون بدیهی است :)

از اونطرف قرار نیست **رنج** بکشین. از نظر من اگر یاد گرفتن چیزی برام لذت بخش نیست، یعنی بهتره یادش نگیرم چون یاد گرفتن چیزی که لذت بخش نیست به معنی استفاده از چیزی است که لذت بخش نیست و شاغل شدن در چیزی که لذت بخش نیست. خب چه کاریه! اگر من از پایتون خوشم نمی یاد یا آر دوست ندارم یا اصولا کامپیوتر اذیتم می کنه، خب می رم مربی تنیس می شم!

این رو هم اضافه کنم که من شخصا برای یاد گرفتن هر چیزی اول یک مساله تعریف می‌کنم، بعد با عبارت هایی مثل `X tutorial` در اینترنت جستجو می کنم و چند راهنمای انگلیسی رو باز می کنم و همه رو یک نگاه می کنم و انتخاب می کنم که کدوم رو می خوام بخونم و تا آخر می خونمش و بعد شروع به کار با X می کنم تا کارم تا حدی پیش بره. اگر در این مرحله مطمئن شدم که می خوام X رو ادامه بدم، یک کتاب می گیرم و می خونم و بعد اگر برام جذاب بود پروژه های جدید بر می دارم و کتاب های جدید. این روزها دیدن ویدئوهای آموزشی هم بسیار مفیده و مثلا برای بیگ دیتا یا یادگیری ماشینی من حجم زیادی ویدئو دارم که وقتی بیکارم توی خونه می ذارم پخش بشه و زیر چشمی نگاهشون می کنم و هر جا لازم بشه دقتم رو بیشتر

می کنم.

شناور شدن در فضا بسیار مفیده. اگر می خواین زبان یاد بگیرین، اخبار رو فقط به اون زبان بخونین. اگر R یاد می گیرین، وبلاگ‌ها و سایت‌هایی که جواب سرچ‌هاتون هستن رو به فیدخون اضافه کنین و همیشه نگاهش کنین. عضو جامعه باشین تا ببینین بقیه چیکار می کنن تا ایده‌ها و کارهای جدید و جنبه‌هایی که باهاش برخورد نداشتین رو هم ببینین. با اینکار جلوی ایزوله شدن خودتون - و در نتیجه حس اشتباه خود خفن پنداری - رو می‌گیرین. آخرین توصیه هم ساخت صحیح زیرساخت‌ها است. اگر قراره من آمار استدلالی یاد بگیرم لازمه زیرش (مثلا آمار توصیفی یا ریاضیات پایه) رو به خوبی بلد باشم و اگر قراره داده‌های بزرگ یاد بگیرم لازمه مثلا با HDFS آشنا باشم و کوچکترین نفهمیدن زیرساخت باعث می شه چیزهایی که بعدا یاد می گیرم قرص و محکم نباشن و شکسته بسته بشن. جالبه که یاد گرفتن درست لینوکس به نظر من بیشتر از دو ماه طول نمی کشه ولی خیلی ها هستن که سال‌ها با لینوکس کار می کنن بدون اینکه این دو ماه رو وقت بذارن و ابزارشون رو دقیق بشناسن. در **یاد گرفتن یک چیز** باید واقعا حواسمون باشه که اون چیز رو یاد بگیرم و فقط دنبال **راه انداختن کار با یک چیز** نباشیم. معمولا یک حرفه ای و یک غیرحرفه‌ای هر دو می تونن کارهای روتین رو راه بندازن ولی کسی سینیور / حرفه ای می شه که درک می کنه اون پایین چی می گذره.

از یادگرفتن لذت ببرین و سعی کنین هیچ کاری رو بدون اینکه درک کنین دارین چیکار می کنین نکنین. با این روش خیلی زود در چیزی که دارین انجامش می دین حرفه ای می شین.

۱۳.۵ ایجاد انگیزه و تمرکز

ما کارهای خوبی رو دوست داریم بکنیم. ما به شکل سنتی در شروع هر سال فهرستی کاغذی یا ذهنی از کارهای اون سال داریم. ما می‌دونیم می‌خوایم وقتی بزرگ شدیم چی بشیم و ما می‌دونیم که قدم بعدی خوب در این روز اینه که چی رو یاد بگیریم. اما تقریبا هیچ کدوم از ما فرانسه یاد نمی‌گیریم، قهرمان هک نمی‌شیم، سیکس پک نداریم و ...

بقیه هم این وضعیت رو می‌دونن. باشگاه‌ها می‌دونن که می‌تونن چند برابر ظرفیت ثبت نام کنن چون اکثر کسانی که ثبت نام کردن دو سه جلسه بیشتر به باشگاه نخواهند اومد. سایت‌های تخفیف گروهی و غیره سود می‌کنن چون درصد قابل توجهی از کسانی که پول می‌دن تخفیف‌ها رو می‌خرن حوصله نمی‌کنن برن از تخفیفی که براش پول دادن استفاده کنن.

چرا اینطوریه؟ گفتن اینکه همیشه اینطور می شه چون آدم انگیزه‌اش حفظ نمی‌شه یا پشتکار نداره مثل اینه که بگیم یک خودرو در مسیر متوقف شده چون سرعتش به صفر رسیده! خب سوال اصلی اینه که چرا انگیزه نداریم و چرا پشتکارمون حفظ نمی‌شه. به نظر من اینها اصلی‌ترین عواملی هستن که باعث می شن پشتکار ما در طول مسیر از بین بره و نظرات‌مون تغییر کنه و تمام سال‌های بعدی رو هم مثل سال‌های قبلی بگذرونیم:

پیشرفت سخت است

فانتزی اینکه ما یک قهرمان باشیم خیلی جذابه ولی مسیر اینکه به قهرمانی برسیم مسیر سخت و پر حوصله‌ای است. برای یک دونده دوی استقامت دائما فکر کردن به اینکه «پس کی می رسم» بدترین کار است. من همیشه نقل می کنم که «آسانسور پیشرفت اختراع نشده و باید قدم به قدم پیش رفت». اگر قراره واقعا یک روز نفر اول فلان چیز باشیم، لازمه که هر روز یک قدم پیش بریم بدون اینکه نگران باشیم که خیلی ها از ما جلوتر هستن. در واقع شما فقط وقتی می تونین در دنیاهای کامپیوتری و خیلی چیزهای دیگه زندگی بهترین باشین که از مسیر لذت ببرین نه از بودن در هدف. درست همونطور که یک کوهنورد باید از کوهنوردی کردن لذت ببره نه از «بودن در قله».

از خودمون نباید جلو بیافتیم

من در مدرسه تیزهوشان درس می‌دادم و به نظرم بدترین چیز در اونجا این بود که بعضی ها از خودشون جلو می‌افتادن. گروهی بود که به همراهی «مربی»هاشون در دبیرستان کارت صوتی کامپیوتر ساخته بودن و خب معلومه که قدم بعدی این آدم نمی تونه «یاد گرفتن الفبای الکترونیک» باشه. اگر شما دارین برای بقیه جلوه‌ای «جلوتر از خودتون» نمایش می‌دین، منطقا هیچ گاه وقت و حوصله برداشتن قدم‌های واقعی اولیه رو نخواهید

داشت. برای روشن شدن مساله یک مثال می‌زنم. فرض کنین گروه از دوستان شما تصمیم گرفتن برن استخر و شنا یاد بگیرن. شما هم دوست دارین قهرمان شنا باشین و برای اینکه قافله عقب نمونین وقتی از استخر برگشتن، بهشون می گین که درسته نیومدین استخر ولی شناتون خیلی خوبه. این باعث می شه دیگه هیچ وقت نتونین باهاشون برین استخر و حتی پایه‌های شنا رو یاد بگیرین. اگر قراره در دنیای تکنولوژی پیشرفت کنین باید همیشه سعی کنین زیرپاتون سفت باشه، از خودتون جلو نیافتین و هر چیزی که متوجه نمی‌شین رو به شکل پایه‌ای بخونین و یاد بگیرین. اتفاقا آدم های باسواد اطراف شما کسانی هستن که مقدمات رو خوب بلدن نه کسانی که چیزهای پیشرفته رو خیلی خوب بلدن (:

مسیر حرکت شما باید رو به جلو باشه

همین الان که دارین این رو می‌خونین به گذشته فکر کنین و ببینین برای پیش اومدن واقعا چه قدم‌هایی برداشتین؟ اگر من سه ساله که فرانسه نخوندم نمی‌تونم هدف سال آینده رو بذارم «سه تا زبان یاد بگیرم». باید در مسیر معقولی آهسته ولی منظم حرکت کرد. هیچ کس نمی‌تونه از فردا یک آدم دیگه بشه! اهداف کوچیکتر انتخاب کنین وآروم آروم بهشون برسین.

مواظب سرمایه‌داری خبیث باشین

برای اینکه سیستم سرمایه داری بچرخه، ما باید دائما یکسری آرزو بخریم و بفروشیم. یکی مشغول فروختن افزایش دهنده سایز است و یکی دیگه مشغول فروختن آپارتمانی در ترکیه که قراره توش زندگی رویایی داشته باشین و یکی دیگه سعی می کنه خدای فلان چیز بشه تا همه بهش احترام بذارن یا دوستش داشته باشن یا پولدار بشه! دنیای واقعی جایی نیست که ما بتونیم به همه آرزوهامون برسیم؛ بخصوص که همین که به اولین آرزو برسین سه تا آرزوی جدید جلوتون سبز می‌شه. در حین نوشتن این من توی یک کافه نشسته‌ام. از اطرافیان می‌پرسیم نظرشون در این باره چیه. اولی صاحب کافه است که روزی آرزوش بوده کافه خودش رو داشته باشه ولی امروز چندان هم شاد نیست. دومی یک دانشجوی دکترا است که بچه و زندگی اش رو داره؛ زمانی که مهندس بوه آرزوش این بوده که روزی دکترای جامعه شناسی باشه ولی هنوز شاد نیست و نفر سوم پزشکه و شرایطش نسبتا مشابه.

واقعا شما هر چقدر هم که پیشرفت کنین بازم حس می کنین موفق نشدین. همین الان من نگران هستم که چرا در تابستان پیش رو سیکس پک ندارم و اگر داشتم بهتر بود و لازمه عاقلانه به خودم بگم که من اصلا اون آدم نیستم (: شما هم مواظب باشین، شاید دارین به اندازه کافی پیشرفت می‌کنین ولی متوجه‌اش نیستین.

شما تعیین کننده همه چیز نیستین

آدم ها شرایط بسیار متفاوتی دارن. مغز ما با هم فرق می‌کنه و اینکه شما الان علاقمند به تکنولوژی هستین معنی اش این نیست که بهترین در دنیای تکنولوژی خواهید بود. من سال‌ها به شکلی جدی یادگیری و تمرینات فوتبال رو دنبال می‌کردم ولی من اصولا تیپ ورزشکار نیستم. این رو باید قبول داشت. از اونطرف من از شش سالگی با کامپیوترها بزرگ شدم در حالی که یک نفر شاید اولین کامپیوترش رو بعد از اینکه یکسال از کار کردنش توی یک کارگاه گذشت بتونه بخره. شرایط آدم ها فرق می کنه و این ایده که «هر کس تلاش کنه موفق می‌شه» تا حد زیادی غیرواقعی است. مثال پرت کردن کاغذ توی سطل عالیه:

معلمی در سر کلاس از بچه‌ها می‌خواد نفری یک کاغذ بردارن، اسمشون رو روش بنویسن و بعد مچاله‌اش کنن. بعد سطل آشغالی رو پای تخته می‌ذاره و می‌گه هر کس بتونه از جایی که نشسته کاغذش رو داخل سطل بندازه، در زندگی موفق / پولدار / مشهور و ... خواهد بود. نفرات جلویی با درصد موفقیت بیشتری پرت می‌کنن و اصلا حواسشون نیست که عقبی‌ها شانس خیلی کمتری دارن. کاملا می‌شه پذیرفت که کسی که در عقب کلاس بیشتر تلاش و تمرین و پیگیری و .. کنه شانسش بالا می‌ره ولی در نهایت جایی که شما نشستین اولین تعیین کننده است و درصد موفق‌ها در آخر کلاس در پایین‌ترین مقدار ممکن. این مثال در دنیا هم صادقه. از اونطرف هم شما با یک سیستم بزرگ طرف هستین که سعی می کنه به شما جهت بده. درس‌هایی که بخونین اونجا تعیین می‌شه، اینکه دسترسی شما به اینترنت به چه شکل باشه اونجا تعیین می‌شه و ... حتی در سطح متوسط هم شما با خانواده درگیر هستین، با خواسته‌های دوستان، با پدر و مادر، با عشقی که یکهو پیش می یاد و این تیپ چیزها. پس نقش شخص شما در این انتخاب‌ها اونقدرها هم زیاد نیست و باید حواستون باشه که اگر قراره به چیزی که می‌خواین برسین، باید خیلی جدی تلاش کنین و حاضر باشین با عوامل بیرونی هم بجنگین.

انسان یک پیوستار نیست

تصور اشتباه ما از انسان یا حتی «خودم» اینه که در یک پیوستار ثابت زندگی می‌کنیم در حالی که باید بگم اینطور نیست. مغز و بدن ما شدیدا تحت تاثیر هورمون‌ها و شرایط بیرونی و درونی است و مغز شمای گرسنه با شمای سیر مغزی متفاوت است و در نتیجه اگر کسی تصمیم‌های جادی خسته از یک شب پر از نوشیدنی و رقص عالی رو با مغز جادی بعد از خوندن کلی خبر تکنولوژی باحال مقایسه کنه به دو موجود کاملا متفاوت می‌رسه. اولی می‌خواد بازم شادی کنه یا بخوابه و دومی می‌خواد هر طور شده خودش رو برسونه به یک میکروفون و رادیو گیک[1] ضبط کنه. مثال دیگه ورزش است. من وقتی ورزش می‌کنم بدنم پر از هورمون‌های لذتی می‌شه که منو مطمئن می‌کنه فردا هم حتما ورزش خواهم کرد ولی وقتی فردا می‌شه اون هورمون‌ها دیگه توی رگ‌های من نیستن و به نظرم عجیب‌ترین کار این می‌یاد که الان که از تخت‌خواب بیرون بیام و بالا پایین بپرم!

[1] jadi.net

اشتباه ما معمولا اینه که فکر می کنیم «من امروز، من فردا هستم» و دقیقا برای همینه که امروز تصمیم‌های مهمی می‌گیریم که انگار روی سنگ حک شدن ولی فردا کلا بیخیال اون تصمیم هستیم و اولویت‌های دیگه‌ای داریم. خوندن رمان و فلسفه برای همین توصیه می شه. این چیزها شاید به شکلی عمیق تر از یکسری هیجان آنی یا تصمیم منطقی بتونن شیوه تفکر مغز رو شکل بدن و باعث بشن در مواقع مختلف نگاه مغز به دنیا ساختار ثابت‌تری داشته باشه و مثلا همیشه بدونه که ادامه دادن یک مسیر نتیجه بهتری از زیگزاگ رفتن می‌ده.

چیزهای ضروری در مقابل چیزهای مهم

توجه کنید که این حرف‌ها اصلا به این معنی نیست که «امکان نداره» بلکه دقیقا ماجرا برعکسه. آدم‌های موفق تونستن به این عوامل و عوامل مشابه غلبه کنن و به نظر من اگر هدف درست و معقول انتخاب بشه حرکت به سمتش ساده است. چیزی رو انتخاب کنین که براتون مناسبه، واقعا می‌خواینش و از همه مهمتر از مسیر رسیدن بهش هم لذت می‌برین. اگر دنبال شهرت یا پول هستین به هیچ وجه سراغ تکنولوژی و غیره نرین چون مسیر کم‌شانس و سختی است اما اگر واقعا از خود مسیر یاد گرفتن لذت می‌برین، مثل یک کوهنورد قدم به قدم پیش برین و آرزو کنین دیرتر به قله برسین چون خود حرکت است که باید لذت بخش باشه.