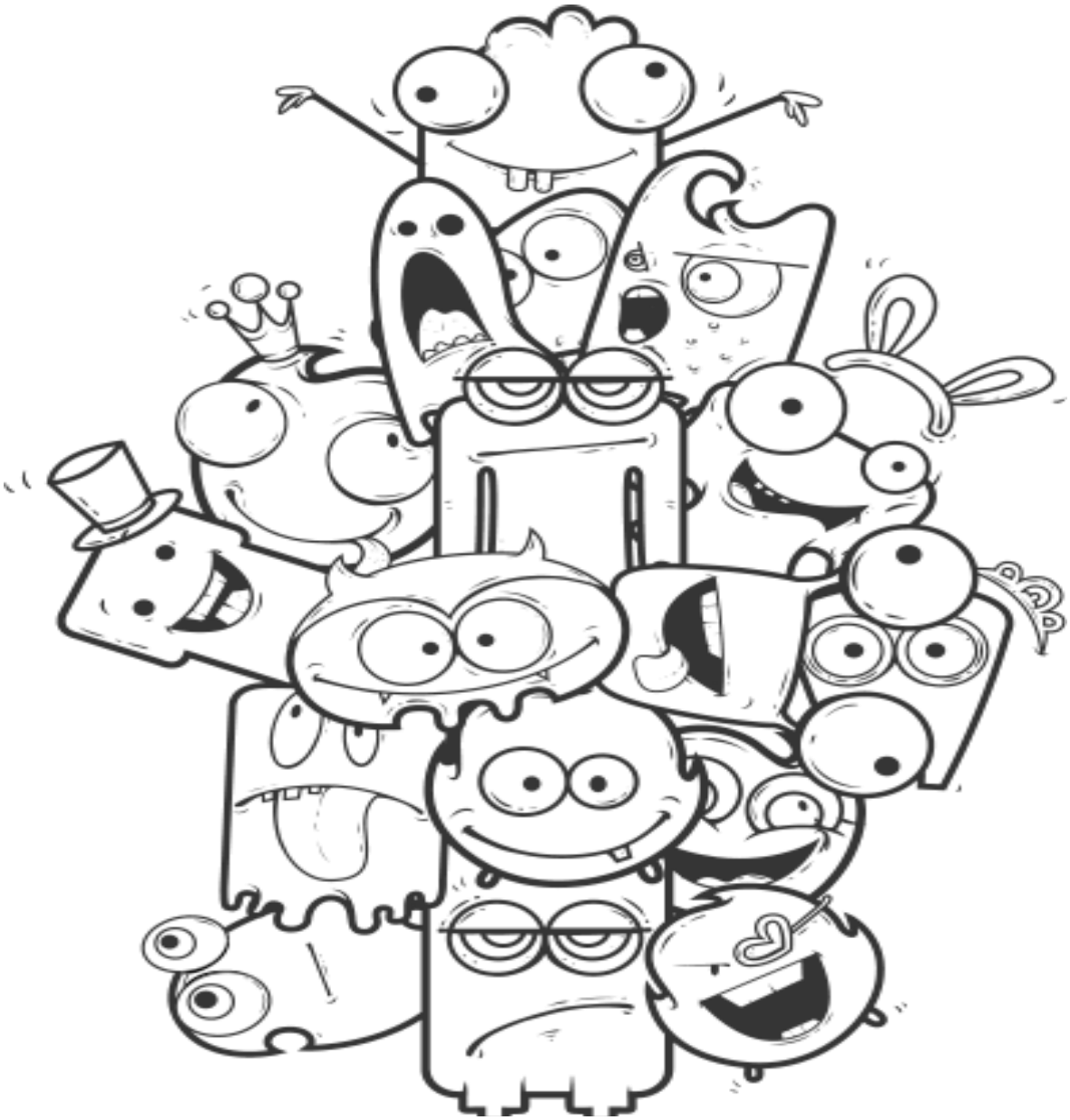


Dance with Linux 

Learn GNU/Linx the fun way

SaharShaker.com

تقدیم به همه کسانی که لینوکس و فلسفهٔ آزادی را دوست دارند و
برای یادگیری و از آن مهم‌تر برای یاد دادن وقت می‌گذارند.

سلام :)

این pdf صرفا دست‌نوشته‌های دختری است که علاقه زیادی به دنیای لینوکس و پایتون دارد و تلاش می‌کند با انتشار این نوشته، افراد بیشتری را با این دنیای هیجان‌انگیز آشنا کند.

برای دریافت آخرین آپدیت در ایمیل‌تون، اینجا کلیک کنید.

saharshaker.com

صرفا جهت اطلاع!

این پی‌دی‌اف به صورت رایگان ارائه شده اگر جایی دیدین که دارن ازش سوء استفاده می‌کنن یا یجورایی می‌فروشنش خبر بدین بیل بیارم با هم بکشیمشون.

خیلی مرسی!

حمایتتون به پیشرفت این کار خیلی کمک می‌کنه.

بزرگترین حمایت شما می‌تونه این باشه که کتاب رو به هر شکلی که فکر می‌کنین share کنین یا معرفیش کنین

کانال تلگرام:

@saharshaker

من فقط بلدم از موس و کیبورد و خود کامپیوتر استفاده کنم.
این کتاب بدردم می‌خوره؟

بله، این کتاب قدم به قدم، لینوکس را معرفی می‌کند و تماما سعی شده مباحث به سادگی بیان شود. با این حال اگر سوالی داشتید، در سطح اینترنت بسیاری از انجمن‌ها و سایت‌ها برای پاسخ به این سوالات شکل گرفته‌اند.

از همین الان سوالاش شروع شد. آخه اگه کسی قبلا سیستم‌عاملی کار نکرده باشه که به فکر لینوکس نمی‌افته. فقط خواهشا با این سوالات جایی نگو فامیل منی. مرسی اه.

چرا نمی‌ذارییییییید من سوالمو بپرسم؟؟؟؟ مسئولین چرا پیگیری نمی‌کنن؟؟؟

یه کلام از مادر عروس، خودش این شکلیه عروس چه شکلیه آخه. مسئولین اگه رسیدگی بخوان بکنن اول دختر تو رو فیلتر می‌کنن. بعدشم کی جلوتو گرفته؟؟؟ داد می‌زنی همش. بگو یا جواب می‌ده یا نمی‌ده دیگه

چرا اصلا لینوکس؟

لینوکس یکی از مهم‌ترین پیشرفت‌های تکنولوژی در قرن 21 است. علاوه بر اینکه به رشد اینترنت کمک بسیار شایانی کرده است، یک الگوی جا افتاده است برای اینکه شرکت‌ها و حتی افراد به تنهایی بتوانند ایده‌های خودشان را پیاده‌سازی کنند.

اندروید بر پایه لینوکس است و همان طور که شاهدیم روزانه هزاران گوشی اندروید به فروش می‌رسد. خود شرکت گوگل از هزاران هزار سرور لینوکسی استفاده می‌کند. با فراگیرشدن اینترنت اشیاء، استفاده از لینوکس بیشتر و بیشتر می‌شود به طوری که روزانه چیزی بیش از 700هزار دستگاه تلویزیون ارائه می‌شود. بسیاری از شرکت‌های معروف و مطرح از لینوکس استفاده می‌کنند و برای پیشرفت و توسعه آن سرمایه‌گذاری می‌کنند، مثل IBM یا Oracle.

حتی لینوکس در بخش‌ها نظامی هم حرف برای گفتن دارد. مثلا از آن در تفنگ‌های هوشمند استفاده می‌شود. تفنگ‌های هوشمند بر اساس یک فاصله مشخص، روی یک هدف تمرکز می‌کنند و آن‌را تعقیب کرده و در زمان مورد نظر به آن شلیک می‌کنند.

در صنایع حمل‌ونقل هم از لینوکس استفاده می‌شود؛ مثلا در قطارهای فوق سریع یا جاهایی که بحث رانندگی خودکار مطرح است.

و یک دنیا استفاده دیگر دارد که به راحتی در اینترنت می‌توانید پیدا کنید، مثل صنایع غذایی، سیاست و...

راستی پرامون کامنت بذارید و مارو از حال خودتون پی‌خبر نذارین!

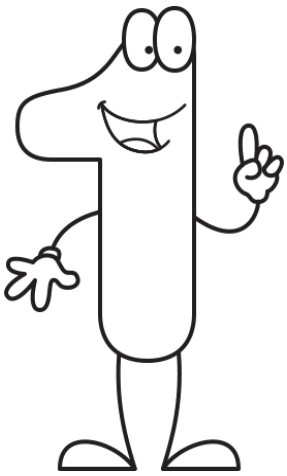
در ضمن نظراتتون رو با گوش جان می‌شنویم کلی پرامون دلگرمیه.

Hey guys, it’s john

یه ربات هم گذاشتیم برای آدمای خسته‌تر از خودم. به هر دلیل اگر حال نداشتین تا سایت برین و احیانا دیدید جایی سوتی دادیم، قبل اینکه آبرمون در سطح محلی، استانی، کشوری و بین‌المللی بره به این ربات تلگرام بفرستید، تو سایت هم نوشتید که چه بهتر

@saharshakerbot

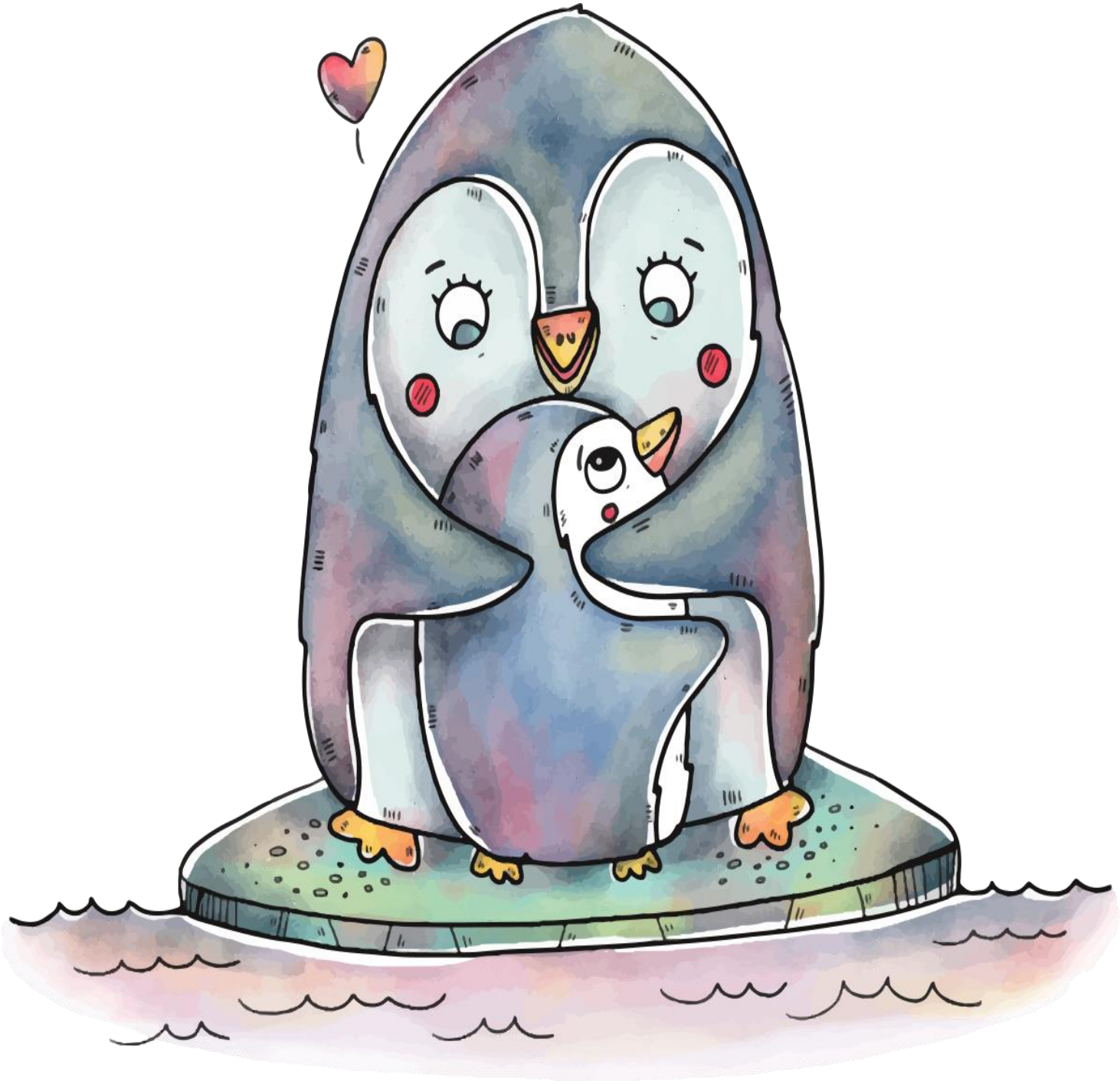
فهرست



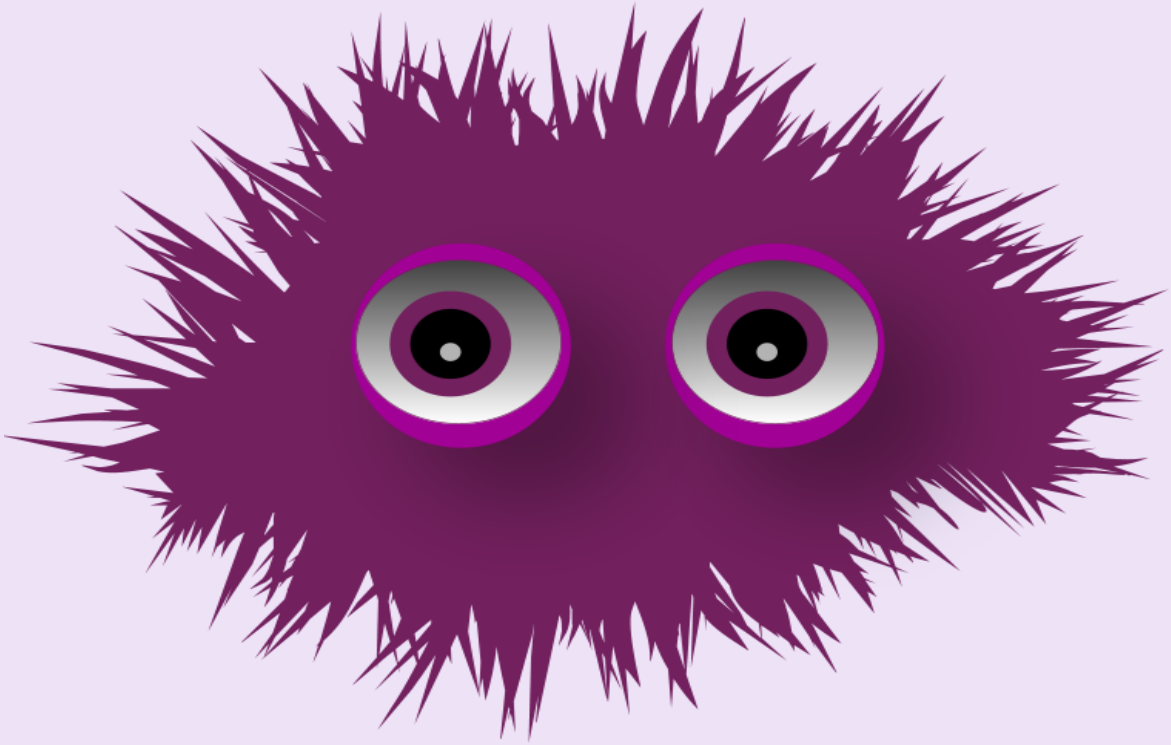
فصل اول

تولد گنو/لینوکس از کجا آغاز شد؟

نصب لینوکس



حال ندارم بخونم الان



حال نداری؟!

بیا یه داستان بگم برات، خوندن این داستان واجب نیست اصلا، ولی می‌تونه کمکت کنه بیشتر عاشق لینوکس بشی و از فلسفه شکل‌گیریش سر دربیاری.

تازه این pdf هم از آسون شروع می‌شه و کم‌کم فنی می‌شه.

لینوکس هیولا نیست که !

اگر کاربری باشید که تازه پا به دنیای لینوکس گذاشته است، شاید کلی سوال و ایده‌های مبهم در ذهن‌تان باشد که دنبال جواب می‌گردد یا از سایت‌ها و افراد مختلف چیزهایی را شنیده است. این فصل جایی است که کمک می‌کند دنیای اپن‌سورس و لینوکس را از زبان خالقان آن بشنویم و با فلسفهٔ شکل‌گیری‌اش بیشتر آشنا شویم.

شخصیت‌های اصلی

Richard Stallman

ریچارد استالمن

بنیان‌گذار پروژه گنو، بنیانگذار جنبش آزادی نرم‌افزار

Eric Raymond

اریک ریموند

برنامه‌نویس، هکر، نویسنده مقاله کلیسا و بازار، پیشتاز دنیای اپن‌سورس

Larry Augustin

لری آگوستین

مدیرعامل و بنیانگذار شرکت‌های

VA Research ،SugarCRM ،Geeknet

Linus Torvalds

لینوس توروالدز

مهندس نرم‌افزار، دولوپرکرنل لینوکس

Bruce Perens

بروس پرنز

برنامه‌نویس، مدافع جنبش اپن‌سورس، او تعریف نرم‌افزار اپن‌سورس را ایجاد کرد.

داستان شکل‌گیری لینوکس [1]

لینوکس چیه؟

لینوس تورووالدز:

برای توضیح این که لینوکس چیه باید توضیح داد سیستم عامل چیه. و قضیه سیستم عامل اینه که آدم هیچ‌وقت قرار نیست اون رو ببینه چون هیچ‌کس در واقع از سیستم عامل استفاده نمی‌کنه. آدم‌ها از برنامه‌های کامپیوترشون استفاده می‌کنند. و تنها هدف زندگی سیستم عامل کمک به اجرای این برنامه‌هاست. پس سیستم‌عامل هیچ‌وقت خودش کاری انجام نمی‌ده. فقط منتظر می‌شه برنامه‌ها درخواست منبع خاصی رو بکنن. یا یه file از روی دیسک بخوان یا برنامه‌هایی بخوان که اونهارو به دنیای بیرون متصل کنه. اون وقته که سیستم عامل می‌آد وارد عمل می‌شه و سعی می‌کنه برنامه نوشتن رو برای مردم آسون‌تر کنه.

اپن‌سورس چیه؟

بروس پرنز:

اپن‌سورس راهیه که مردم بتونن روی نرم‌افزارها باهم همکاری کنن بدون این که درگیر اون همه مشکلات حقوق معنوی بشن. طوری که هر وقت کسی می‌خواد یه نرم‌افزار بخره

[1] این داستان برگرفته از دیالوگ‌های فیلم Revolution Os است.

مجبوره کلی قرارداد بالا و پایین کنه و با کلی وکیل چک و چونه بزنه . به طور کلی ما فقط می‌خواهیم نرم‌افزار کار کنه. و می‌خواهیم که بتونیم از **کمک‌های دیگران برای تصحیح و ... برخوردار باشیم**. بنابراین بعضی از حقوق و اموال معنوی رو یک‌جورهایی قربانی می‌کنیم. و اجازه می‌دیم تمام دنیا از نرم‌افزار استفاده کنن.

بیاید یکم برگردیم عقب؛ قبل از اینکه حرف از لینوکس باشه، یه ریچارد استالمن بود که بابای جنبش نرم‌افزار آزاد لقب داشت.

ریچارد استالمن:

من سال 1971 وارد آزمایشگاه هوش‌مصنوعی MIT شدم. وارد اجتماع شکوفایی از هکرها شدم. کسایی که عاشق برنامه‌نویسی و عاشق کشف کارهایی بودن که می‌تونستن با کامپیوترها بکنن. اون‌ها یه سیستم عامل درست کرده بودن که همش همونجا نوشته شده بود! عضو تیمی شدم که به ارتقای سیستم عامل ادامه می‌داد و قابلیت‌های جدیدی به اون اضافه می‌کرد. این شغلم بود و من عاشقش بودم. ما همه عاشقش بودیم. به همین خاطر این کار رو می‌کردیم. و اسم سیستممون رو گذاشته بودیم: « the incompatible time sharing system». فارسیش می‌شه :«سیستم اشتراک زمانی ناسازگار» که نشون‌دهنده روحیهٔ بازیگوشیه. که مشخصهٔ هکرهاست. هکرها کسانی هستن که از هوشمندی بازیگوشانه لذت می‌برن. خب اولین مشکلی که برامون پیش اومد وقتی بود که دنیای خارج بهمون فشار آورد که کلمهٔ عبور داشته باشیم. ما روی کامپیوترهامون کلمهٔ عبور نداشتیم. و دلیلش این بود که هکرهایی که اون اول، سیستم رو طراحی کرده بودن متوجه شده بودن که کلمهٔ عبور راهی برای رئیس‌هاست که بتونن تمام کاربرها رو کنترل کنن. و نخواسته بودن قفل و کلیدی برای رئیس‌ها بسازن که بتونن اون‌ها رو کنترل کنن. بنابراین این کار رو نکردن و این مورد رو از قلم انداختن. فسلفهٔ ما این بود که هرکس که پشت

کامپیوتر نشسته باید بتونه هرکاری که دلش می‌خواد بکنه. و کس دیگه‌ای که دیروز اونجا بوده نباید بتونه کاری رو که آدم امروز می‌کنه رو کنترل کنه. وقتی اون‌ها روی یکی از دستگاه‌های MIT کلمه عبور گذاشتند، من و یک مشت از هکرهای دیگه خوشمون نیومد. تصمیم گرفتم یه جور هک شورشی انجام بدم. کشف کردم چطور کلمه‌های عبور رو رمزگشایی کنم. اینجوری با نگاه‌کردن به پایگاه داده که کلمات عبور در اون قرار داشت، می‌تونستم بفهمم هر کسی برای ورود به سیستم چی تایپ می‌کنه. بعد به همون آدم پیغام می‌فرستادم و می‌گفتم:

سلام، می‌بینم که کلمهٔ عبور فلان رو انتخاب کرده‌اید! چطور همون کاری رو بکنید که من کردم؟ یعنی به جای کلمهٔ عبور فقط اینتر رو بزنید؟ خیلی کوتاه‌تره و تازه تایپ‌کردنش خیلی آسون‌تره

و البته با این پیغام من داشتم تلویحا بهشون می‌گفتم که امنیت سیستمشون عملا در حد شوخیه ! (جاست فور فانه)در هر صورت با این هک داشتم اون‌هارو دعوت هم می‌کردم. آخرشم یک‌پنجم کاربران اون کامپیوتر به من ملحق شدن. و به عنوان کلمهٔ عبور فقط اینتر رو می‌زدند.

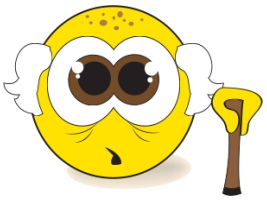
افکاری که الان به چیزی که دنیای اپن‌سورس می‌گیم، از کجا اومدن؟ این کار چطور شروع شد؟ چه کسی شروعش کرد؟

یعنی پیرِ سالای 49 تا 59

بروس پرنز:

این کار در واقع با پیدایش کامپیوتر شروع شد. چون در اون زمان نرم‌افزار بین مردم رد و بدل می‌شد. و فکر می‌کنم تازه در اواخر دهۀ 70 یا اوایل دهۀ 80 بود که مردم شروع کردن به بستن نرم‌افزارهاشون. و گفتن: «نه اصلا نمی‌شه سورس‌کد رو نگاه کنید. نمی شه نرم‌افزار رو تغییر بدید حتی اگه لازم باشه برای کامپیوتر خودتون درستش کنید .در واقع می‌شه مایکروسافت رو مقصر قسمتی از این قضیه دونست. اون‌ها یکی از پیشگامان واقعی مدل نرم‌افزار خصوصی هستن.

همون fix اشُ کنید خودمون!



اواسط دهۀ 70 گروهی از هکرها و کامپیوترکارانِ سیلیکون ولی که تفننی کار می‌کردند، باشگاه کامپیوترهای خانه‌ساز رو تشکیل دادن. (انگلیسیش: Homebrew computer club). سال 76 بیل‌گیتس از شرکت تازه تاسیس مایکروسافت، یه نامه داد به این گروه. تا قبل از این کاربران کامپیوتر بدون اینکه خیلی تو فکر مالکیت نرم‌افزارها باشن آزادانه اون‌رو رد و بدل می‌کردن. بیل گیتس توی این نامه نوشته بود:

برای من حساس‌ترین بازار تفنن در حال حاضر نبود دوره‌های نرم‌افزاری، کتاب‌ها و نرم‌افزارهای خوب است. بدون نرم‌افزار خوب و مالکی که برنامه‌نویسی را بفهمد، کامپیوتر تفننی شما هدر رفته است. آیا کسی برنامه‌هایی با کیفیت برای بازار تفنن خواهد نوشت؟...بازخوردی که ما از صدها نفر که می‌گویند از بیسیک استفاده می‌کنن گرفته‌ایم، بسیار مثبت بوده. هرچند دو نکتهٔ خارق‌العاده وجود داره؛ یک: بیشتر این کاربران بیسیک را نخریده‌اند و دو: با مقدار حق‌التالیفی که ما از این تفننی کاران دریافت کرده‌ایم، دستمزد کاری که صرف آلترا بیسیک شده است کمتر از ساعتی 2دلار می‌شود. آخه چرا؟ همان‌طور که اکثریت تفننی کاران مطلع هستند، بیشتر شما نرم‌افزارهای خود را می‌دزدید پول سخت‌افزار رابابد داد ولی نرم‌افزار را می‌توان شریک شد. چه اهمیتی دارد که کسانی‌که آن‌ها را تولید می‌کنند پولی دریافت کنند؟ آیا این منصفانه است؟ یک کاری که شما با دزدیدن نرم‌افزارها نمی‌کنید جواب دادن به میتس برای مشکلاتی است که داشته‌اید. میتس از فروش نرم‌افزار پولی در نمی‌آورد. کاری که شما دارید می‌کنید این است که جلوی نوشته‌شدن نرم‌افزارهای خوب را می‌گیرید. چه کسی می‌تواند به رایگان کار حرفه‌ای انجام دهد؟

کدام تفننی کار می‌تواند 3سال وقت سر برنامه‌نویسی بگذارد، تمام مشکلات را پیدا کند، مستندات آن را تولید کند و بعد آن‌را به رایگان توزیع کند؟ حقیقت این است که هیچ‌کس غیر از ما سرمایه گذاری زیادی در نرم‌افزارهای تفننی نکرده است. آن هایی که آلتربیسیک را دوباره می‌فروشند چه؟ آیا آن‌ها از نرم‌افزارهای تفننی پول در نمی‌آورند؟ بله ولی کسانی که به ما گزارش‌شده‌اند بالاخره خواهند باخت. آن ها کسانی هستند که تفننی کارها را بدنام می‌کنند و باید از هر جلسهٔ باشگاه که در آن پیدایشان شود بیرونشان انداخت.

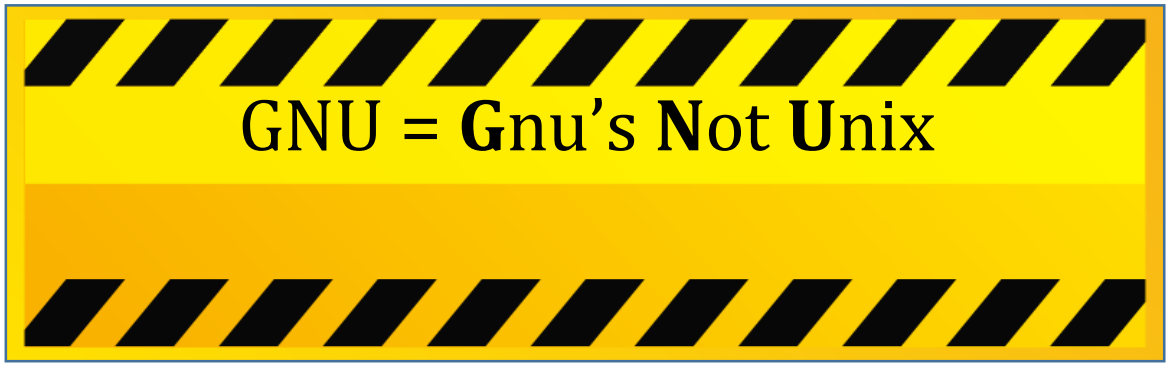
من از نامه‌های هرکسی که بخواهد دِینش را بدهد یا نظر و پیشنهادی داشته باشد استقبال می‌کنم.

بیل گیتس. سهام دار اصلی مایکروسافت.

ریچارد استالمن:

من یه سیستم‌عامل نویس هستم. اگر یک سیستم‌عامل دیگه درست می‌کردم و بعد به عنوان مولف همه رو تشویق می‌کردم که اون رو به دیگران بدن (share کنن) نه تنها می‌تونستم برای خودم راهی فراهم کنم که بدون خیانت به بقیهٔ مردم از کامپیوتر استفاده کنم بلکه می‌تونستم این راه رو در اختیار بقیه هم بذارم. همه راهی برای فرار از اون تناقض اخلاقی پیدا می‌کردن. پس این پروژه رو توی ژانویهٔ 84 شروع کردم. این وقتیه که از شغلم توی MIT استعفا دادم و کار ساخت سیستم‌عامل گنو رو شروع کردم. اینجا باید توضیح بدم که کلمهٔ گنو، خودش یه هکه ! چون مخفف «گنو یونیکس نیست» است.

62 ی



من داشتم سیستم‌عاملی رو درست می‌کردم که شبیه سیستم‌عامل یونیکس بود ولی خود سیستم‌عامل یونیکس نبود. این سیستم فرق داشت مجبور بودیم تمامش رو از صفر بنویسیم. چون یونیکس خصوصی بود. ما اجازه نداشتیم یونیکس رو به اشتراک بذاریم. پس مجبور بودیم جایگزینی براش بنویسیم.

طی دههٔ 80 که ریچارد استالمن داشت پروژه گنو رو می‌ساخت. دانشمندان کامپیوتر توی دانشگاه کالیفرنیا در برکلی در حال ساخت سیستم‌عامل آزاد خودشون بودن. این سیستم عامل که به عنوان یونیکس برکلی یا BSD شناخته می‌شد برپایهٔ هستهٔ

یونیکس و با اجازه AT&T ساخته می‌شد. به هر جهت، به دلیل مشکلات قانونی با AT&T و به دلیل چند پارچگی سورس‌کد ، هکرها و دیگر کاربران غیرسازمانی به کندی BSD را می‌پذیرفتن.

ریچارد استالمن:

خب یونیکس از تعداد زیادی برنامهٔ مجزا تشکیل شده بود که با همدیگه ارتباط برقرار می‌کردن. بنابراین فقط لازم بود این برنامه‌هارو یکی‌یکی جایگزین کنیم. پس شروع کردم به نوشتن. بعد کم کم افراد دیگه‌ای بهم ملحق شدن چون من اطلاعیه‌ای منتشر کرده بودم که از مردم دعوت می‌کرد به من ملحق بشن و توی نوشتن این برنامه‌ها کمک کنن. تا حدود سال 91 عملا همه برنامه‌هارو جایگزین کرده بودیم.

این برنامه‌هایی که جایگزین کرده بودید چه برنامه‌هایی بودند؟

خب ما باید یه سیستم عامل کامل می‌داشتیم. یک کرنل لازمه، که برنامه‌ای که منابع رو به همهٔ برنامه‌های دیگه تخصیص می‌ده، یه کامپایلر لازمه تا برنامه هارو از سورس‌کدی که برنامه‌نویس‌ها می‌فهمن به یک سری عدد ترجمه کنه . برای این کار برنامه‌های دیگری هم در کنار مترجم لازم‌اند. یه دیباگر، یه تکست ادیتور،تکست فورم، ایمیل و... چیزهای خیلی خیلی زیادی لازمه. سیستم عامل‌های شبیه یونیکس صدها برنامه مثل این دارند.

مایکل تیمن:

اطلاعیه ریچارد استالمن رو دیدم و فوریه 87 خودشو ملاقات کردم. اون برای یه دوره آموزشی پنج روزه برای برنامه Emacs به شرکت ما اومد. در طول روز اون درباره روش‌های جدید نگاه به Emacs توضیح می‌داد. و روش‌های گسترش دادن و بهترکردن سورس‌کدهای Emacsحرف می‌زد. ولی شب‌ها سرش حسابی گرم یه کامپایلر بود و

اون‌موقع هنوز اون رو برای عموم منتشر نکرده بود. برای همین یکم مراقب بود که چه کسی داره سورس‌کدش رو می‌بینه. ولی من خیلی مشتاق بودم و تا ماه ژوئن اون سال که اعلامش کرد و بلافاصله از اینترنت دانلودش کردم با برنامه ور رفتم و چندتا سوال ازش پرسیدم وقتی سورس کد رو براش پس فرستادم، اون با ناباوری دید که چقدر سریع تونسته بودم تکنولوژی اون‌رو ارتقا بدم.

وقتی از آزادی حرف می‌زنیم دقیقا از چی حرف می‌زنیم؟!

ریچارد استالمن:

مهم ترین خاصیت گنو اینه که نرم‌افزار آزاده. و آزاد بودن نرم‌افزار نه به قیمتش که به آزادی ربط داره. پس به آزادی بیان فکر کنید. آزادی ای که دارم صحبتش رو می‌کنم اینه که اگر خواستین تغییراتی در نرم‌افزار بدین یا کسی رو استخدام کنید که براتون تغییرات رو انجام بده. اگر برای کسب وکارتون از یه نرم افزار استفاده می‌کنید که بتونین مجددا نسخه‌های اون رو کپی کنید و با مردم share کنید. تا برنامه رو بهتر کنید و نسخهٔ بهتر شده رو منتشر کنید که بقیه مردم هم از مزایای اون بهره‌مند بشن. و این آزادی هاست که نرم‌افزار آزاد رو از نرم‌افزارهای غیرآزاد متمایز می‌کنه. این آزادی‌هاست که مردم رو قادر می‌کنه یه اجتماع (یه کامیونیتی) تشکیل بدن. شما اگر تمام این آزادی‌ها رو نداشته باشید بینتون تفرقه می‌افته و یکی بهتون مسلط می‌شه .

نرم افزار آزاد که می‌گین همه چیش ولنگه و بازه؟!!هیچ لایسنس یا مجوزی نداره؟

اگه نرم‌افزارهارو در مالکیت عمومی بگذاریم یکی می‌تونه بیاد و یک خرده تغییر توی اون بده و تبدیلش کنه به یک بستهٔ نرم‌افزاری خصوصی و معنیش اینه که کاربرانی که از نرم‌افزار ما استفاده می‌کنند آزادی همکاری یا به دیگران دادن اون رو نخواهند داشت. برای جلوگیری

از این وضعیت ما از تکنیکی به اسم "کپی لفت" استفاده می‌کنیم. مفهوم کپی لفت یه "کپی رایت" سر و تهه. کاری که ما می‌کنیم اینه که می‌گیم این نرم‌افزار کپی رایت داره و ما نویسندگان اون به شما اجازه می‌دیم که دوباره اون رو **توزیع کنید، تغییر بدید یا چیزی بهش اضافه کنید.** ولی وقتی که دوباره اون رو توزیع می‌کنید باید دوباره تحت همین شرایط باشه. نه بیشتر نه کمتر.

تا هرکسی که اون رو از شما می‌گیره آزادی همکاری با بقیهٔ مردم رو، اگه بخواد، داشته باشه. و به این روش، نرم‌افزار هر جایی که بره آزادی همراهش می‌ره. و همکاری با بقیهٔ مردم. و تشکیل اجتماعات به یک حق لاینفک تبدیل می‌شه.

این چی بود؟ این چه لایسنسی بود؟

خب کپی لفت ایده کلی بود. برای استفاده از اون باید یک نمونهٔ مشخص داشت. نمونهٔ مشخصی که ما برای اکثر بسته‌های نرم‌افزاری گنو به کار می‌بریم این است:

GNU GENERAL PUBLIC LICENSE (GPL)

(مجوز عمومی همگانی گنو)

یک سند به زبان حقوقی که این کار رو انجام می‌ده. خیلی ها همین مجوز رو به کار بردند، مثلا : لینوس تورووالدز هم از این مجوز برای لینوکس استفاده کرد.

بروس پرنز:

خب مجوزی که من بکار بردم، GPL بود همونی که ریچارد استالمن نوشت و به نظر من این مجوز کار بسیار حیرت انگیزیه. این یکی از معدود مجوزهای نرم‌افزاریه که از جایگاه اجتماع نوشته شده. نه از جایگاه محافظت از یک شرکت یا در مواردی مثل مجوزهای MIT

و BSD. از جایگاه اجرای یک برنامهٔ تحت حمایت دولتی . GPL از این نظر واقعا بی‌نظیره. اون فقط مجوز نیست یک فلسفهٔ کامله که فکر می‌کنم انگیزه تعریف اپن‌سورس شد. من کتمان نمی‌کنم که خیلی از کارهای من از کارهای استالمن ناشی شده.

خیلی از شرکت‌های تجاری شروع به استفاده از این نرم‌افزارها و سیستم‌عامل گنو و لینوکس کردند که نقطه شروع آن‌ها، آزمایشگاه تحقیقات الکترونیک در دانشگاه استنفورد بود. و منبع الهام اولین شرکت‌های گنویی و لینوکسی شد.

چجوری از گنو پول درمیارین؟

لری آگوستین:

مایکل تیمن شرکتی تاسیس کرده بود به اسم نرم‌افزاری سیگنوس. با این ایده که حول نرم‌افزارهای آزاد گنو مشاوره و خدمات بفروشه. و خب کار مایکل با سیگنوس خیلی گرفت.

مایکل تیمن:

کلی وقت صرف کردم که بفهمم چجوری قراره پول دربیاریم. در بیانیهٔ اصلی گنو که بخش آخر کتاب راهنمای Emacs گنو بود. استالمن چند راه مختلف برای پول درآوردن پیشنهاد کرده بود.

ریچارد استالمن:

از آغاز جنبش نرم‌افزارهای آزاد تصور من این بود که جا برای تجارت هم در آن وجود داره. یکی از مزایای نرم‌افزارهای آزاد اینه که بازار آزادی برای هر نوع خدمات و پشتیبانی وجود داره. این جوری وقتی آدم در کارش از نرم‌افزار استفاده می‌کنه و پشتیبانی (ساپورت) خوب می‌خواد، می‌تونه آدم‌هایی رو پیدا کنه که این کار رو براش انجام بدن. می تونه شرکت‌هایی رو پیدا کنه که کار تامین پشتیبانی هستند. اون ها عموما مجبورند پشتیبانی خوبی به شما ارائه بدن وگرنه شما کس دیگه‌ای رو پیدا می‌کنید. در مورد نرم‌افزارهای خصوصی ، پشتیبانی انحصاریه، عموما یک شرکت هست که سورس‌کد رو داره و فقط اون‌ها هستند که می‌تونن پشتیبانی ارائه بدن. یعنی عموما آدم به لطف این طرف وابسته است! مثلا در مورد مایکروسافت وضع به همین منواله. به این ترتیب تعجبی نداره که پشتیبانی‌شون اینقدر بده (البته این صحبت ریچارد استالمن مال قدیم بود الانو نمی‌دونم! عموما ما ایرانیا که اصل نرم‌افزار رو استفاده نمی‌کنیم که پشتیبانی بخوایم یا از طرفی هم مدام تحریمیم!!)

مایکل تیمن:

مزایای نرم‌افزارهای آزاد فوق‌العاده بودند، مگر هزینهٔ پشتیبانی داخلی و این مدیران رو خیلی خیلی نگران می‌کرد. بنابراین ایده اساسی که من داشتم این بود که بتونیم مدلی درست کنیم که بتونه 2 تا 4 برابر قابلیت پشتیبانی و کمک‌رسانی رو که یک مهندس داخل سازمان داره در اختیار بذاره. و بتونیم این کار رو با نصف تا یک‌چهارم همون هزینه انجام بدیم. خیالمون می‌تونه جمع باشه که مردم واقعا اون رو میخرن. و تا پاییز اون سال ما همه چیز رو درباره این که توی گروه فنی چه کسانی باید باشند، شرایط فروش چی باید باشه و شاخص‌های کلیدی قیمت چی هستند رو درآورده بودیم. در واقع در نوامبر 89 بود که شرکت رو ثبت کردیم. اونم به اسم سیگنوس.

بروس پرنز:

خیلی دقیق می‌تونم بگم که سیگنوس اولین شرکتی بود که مختص نرم‌افزارهای آزاد بود. پشتیبانی سیگنوس از نرم‌افزارهای آزاد شکاف بسیار مهمی رو پر کرد چون نرم‌افزارهای بسیار خوبی داشتیم که می‌شد مفتی به دستشون آورد ولی نمی‌شد براشون پشتیبانی پیدا کرد. اون ها پولشون رو با ارائهٔ پشتیبانی در می‌آوردند.

ریچارد استالمن:

کرنل آخرین چیزی بود که نوشتنش رو شروع کردیم و هنوز خیلی از شروعش نگذشته بود که سر و کلهٔ لینوس تورووالدز پیدا شد.

اون یک کرنل ساخت و قبل از اینکه ما مال خودمون رو راه بندازیم راهش انداخت و خیلی هم خوب و حسابی راهش انداخت. اسم کرنلش Linux گذاشت.

لینوس تورووالدز:

هدف اولیهٔ من هدف خیلی شخصی‌ای بود. این که بتونم روی کامپیوتر خودم محیطی شبیه به محیطی که روی کامپیوترهای دانشگاه بهش عادت کرده بودم ایجاد کنم و نمی‌تونستم چیزی پیدا کنم که از این جهت به دردم بخوره. این طور شد که من اصولا تمام عمرم رو پشت کامپیوتر نشسته بودم. به این نتیجه رسیدم که باید خودم دست به کار بشم. بیشتر ایده‌ها رو اون اوایل از سیستم عامل سان گرفتم (SunOs) که چیزی بود که اون موقع توی دانشگاه ازش استفاده می‌کردیم. حالا چی شد که من انقدر سریع تونستم راهش بندازم، این بود که از روش برنامه‌نویسی قدیمی که جوابش رو پس داده بود استفاده کردم و مفهوم یکپارچه بودن رو شکستم.

یکپارچه بودن اصولا به این معنیه که سیستم‌عامل یک نهاده و تقسیم ناپذیر. در حالی که در microkernel سیستم‌عامل در حقیقت چیزی نیست جز مجموعه‌ای از

سرویس‌دهنده‌هایی که کارهای مختلفی انجام می‌دن و پروتکل مشترکی برای ارتباط با همدیگه دارن.

خب چطور شد که پروژه گنو که این همه جلوتر بود و از قبل داشت این کار رو می‌کرد چطور شد که اون اینجوری آخر کار وارد شد؟

ریچارد استالمن:

خب در واقع ما کار روی Hurd گنو رو خیلی قبل از اینکه اون لینوکس رو شروع کنه شروع نکردیم.و اتفاقا ما طراحی‌ای رو انتخاب کردیم که طراحی بسیار پیشرفته‌ایه. از نظر قدرتی که به آدم می‌ده ولی از اون طرف معلوم شد که دیباگ‌کردنش هم خیلی سخته. ما تصمیم گرفتیم کرنل رو که یک برنامه بود چند تکه کنیم و به کلی برنامهٔ کوچکتر تقسیمش کنیم که برای ارتباط برقرار کردن پیام‌های ناهمزمان به هم می‌فرستادن. مسئله اینه که این روش برنامه‌نویسی زمینهٔ زیادی برای باگ‌ها داشتن. باگ هایی که پیداکردنشون اغلب خیلی سخته چون به این بستگی داره که آیا فلان برنامه، فلان پیغام رو قبل از این می‌فرسته که اون یکی برنامه بهمون پیغام رو می‌فرسته یا بعدش؟!! و نتیجه این شد که سال‌ها طول کشید که تونستیم برنامه رو به کار بندازیم.

رابطه لینوکس با پروژه گنو چیه؟

لینوس تورووالدز:

خب می‌شه گفت در چندین سطح با گنو رابطه داره. یکیش سطح فلسفیه که آدم فکر کنه بازکردن سورس ایده خوبیه.

ریچارد استالمن:

وقتی لینوس کرنل‌اش رو نوشت این کار رو برای پروژه گنو نمی‌کرد. این کار رو مستقلا انجام می‌داد و مستقلا هم منتشرش کرد. ما هم خبر نداشتیم. ولی بعضیا که خبر داشتن تصمیم گرفتن بگردن ببینن دیگه چی میتونن پیدا کنن تا کنار کرنل‌شون بذارن و یه سیستم‌عامل کامل داشته باشن. اونا گشتن و دیدن که عجب! تمام چیزایی که لازم دارن از قبل وجود داشته . فکر کردن چه شانسی آوردیم! ولی در واقع شانسی در کار نبود. اون ها تمام تکه‌های سیستم گنو رو پیداکرده بودن که فقط کرنل رو کم داشت. به این ترتیب وقتی همه چیز رو سر هم کردن در واقع داشتن لینوکس رو توی شکاف سیستم گنو جا می‌دادن. ولی اونا اینو نمی‌دونستن!

لینوس تورووالدز:

خیلی از این برنامه‌ها کار بنیاد نرم‌افزارهای آزاد و کار آدم‌هایی که لینوکس رو دوست داشتن بود. و یک همزیستی بین لینوکس و این برنامه‌ها وجود داره.که این برنامه‌ها روی لینوکس اجرا می‌شن و در عین حال از لینوکس به عنوان پلتفرم استفاده می کنن. در حالی که لینوکس به خاطر امکان استفاده از این برنامه‌ها از اون‌ها فایده می‌بره!

چه برنامه‌هایی؟

در واقع مهمترینشون کامپایلر سی گنو است. بدون وجود کامپایلر سی ، تولد لینوکس و اکثر این پیشرفت‌ها شدنی نبود. لینوکس از GPL استفاده می‌کنه و من با فلسفه‌ای که پشت GPL هست موافقم. این رو هم بگم که GPL خودش اصلا متن قشنگی نیست. دلیلش هم احتمالا اینه که هیچ متن حقوقی‌ای اون قدرها خوشایند نیست و نمی‌تونه هم باشه.

لری آگوستین:

کم کم متوجه شدم که تو خونه به یه دستگاه یونیکس احتیاج دارم. رفتم دنبالش و دیدم می‌تونم از لینوکس روی یه کامپیوتر شخصی استفاده کنم. با حدود 2000 دلار یه سیستم سرهم کردم که 1/5 تا 2برابر سریع‌تر از سیستم‌های sun بود. فوق العاده بود. من 1/5 تا 2برابر سرعت رو با 1/3 تا 1/4 قیمت داشتم! یه چیزی تو مغزم جرقه زد! فهمیدم اینجا فرصتی نهفته است. این شانس ما بود که کاری رو واقعا بهتر از sun انجام بدیم اون هم با استفاده از اپن‌سورس و لینوکس.

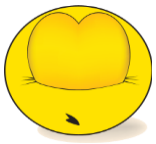
ایده اسم لینوکس چی بود؟

لینوس تورووالدز:

لینوکس اسمی بود که پیش خودم روش گذاشتم فقط به خاطر اسمم لینوس و x هم که حتما باید باشه بخاطر یونیکس. اولش فکر کردم نمی‌تونم اسمش رو رسما بذارم لینوکس چون خیلی خودخواهانه می‌شه.

ریچارد استالمن:

اون ها فکر کردن یه عالمه برنامه پیداکردن و دارن اون رو دور لینوکس می‌ذارن اینجوری شد که اسم قضیه رو گذاشتن سیستم عامل لینوکس. و این اصطلاح جا افتاد و نتیجه این شد که میلیون‌ها نفر حالا دارن از این گونهٔ سیستم گنو به اسم سیستم‌عامل گنو/لینوکس استفاده می‌کنن و اکثرشون این رو نمی‌دونن.

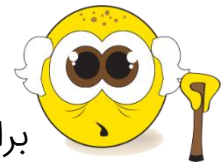


بعضی‌ها پیشنهاد می‌کنن این سیستم گنو/لینوکس نامیده بشه. نظر شما چیه؟

لینوس تورووالدز:

خب من فکر می‌کنم موجهه، ولی در صورتی موجهه که یه توزیع گنویی از لینوکس منتشر کنند. همونطور که لینوکس رد هت Red Hat Linux مشکلی نداره یا لینوکس سوزه SuSe Linux یا لینوکس دبین Debian Linux. چون در واقع اگر آدم بخواد توزیع خودشو از لینوکس درست کنه اسمش رو هم خودش می‌ذاره. ولی اینکه کلا اسمش رو بذاریم گنو لینوکس به نظرم مسخرست.

چیزی که باعث شد لینوکس فراتر از دنیای برنامه‌نویسا بره

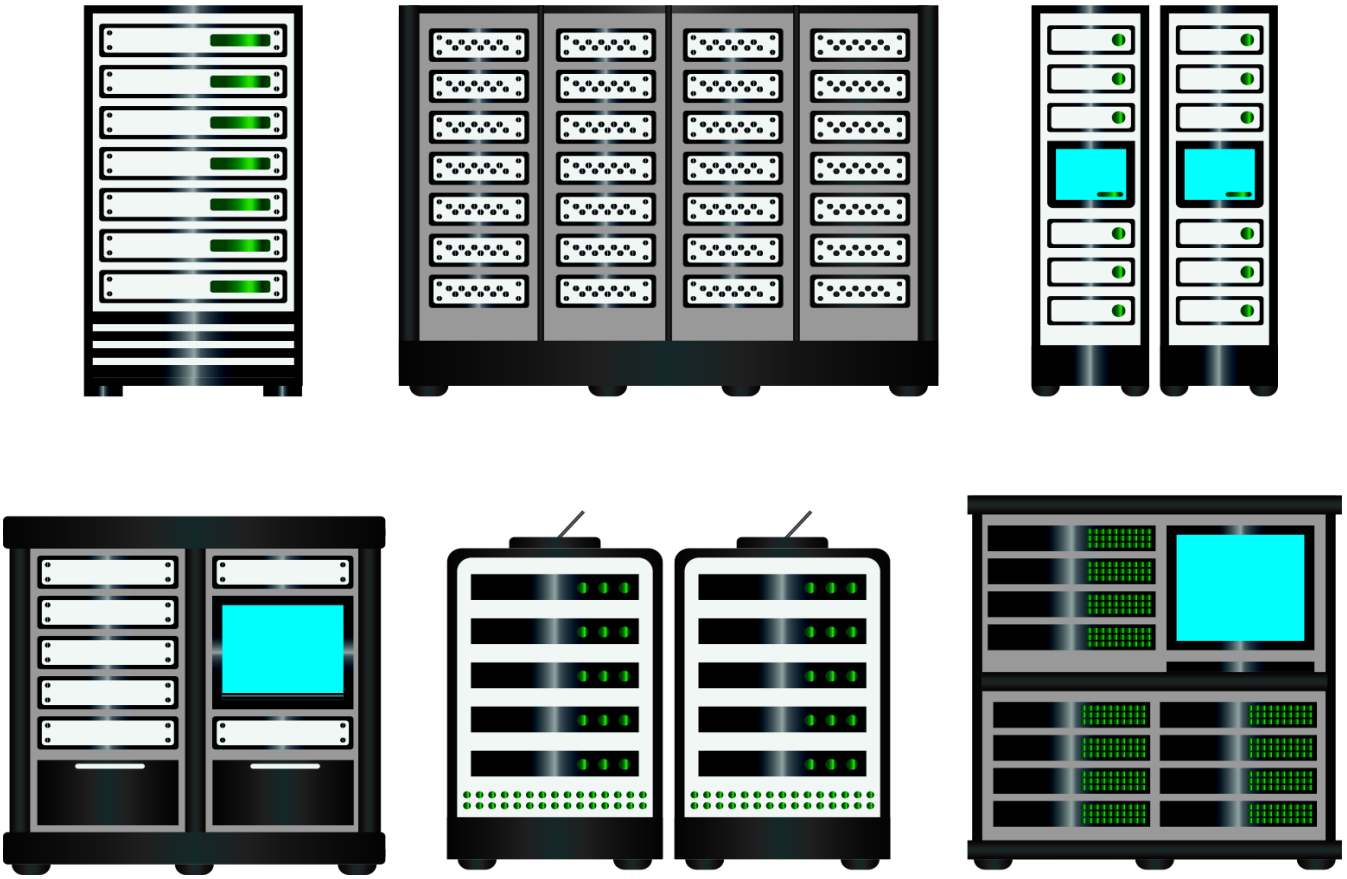


برای رشد لینوکس به ماورای دنیای برنامه‌نویسان کامپیوتر ، استفاده و کاربردی نیاز بود که اون رو به یه فناوری غیرقابل اغماض بدل کنه چیزی که باعث عبور از این آستانه شد به وجود آمدن برنامه‌ای بود که ساختن وب‌سایت‌های پیچیده رو امکان پذیر می‌کرد. این برنامه آپاچی وب‌سرور بود. Apache web server

اریک ریموند:

گل سرسبد برنامه‌های لینوکس بدون شک آپاچی وب سرور بود. اگر به تاریخچهٔ لینوکس دقت کنید می‌بینید که منحنی استفاده از لینوکس و منحنی استفاده از اینترنت دقیقا همدیگه رو دنبال می‌کنن. سال 93 که پروژه آپاچی وب سرور شروع شد همون سالی بود

که انفجار پر طرفدار ISP هم شروع شد. وقتی که اینترنت یک کالا فروشی عمده شد. و ایده تجارت الکترونیکی مبتنی بر وب و ارتباطات عمومی به حقیقت پیوست.



BRIAN BEHLENDORF

فکر می‌کنم آپاچی یکی از اولین برنامه‌هایی بود که باعث شد مردم بگن : خب اگر لینوکس نصب کنم یه فایده مشخصی برام داره. البته اون موقع کلی برنامهٔ جالب روی لینوکس وجود داشت.ولی می‌دونید واقعا انگیزه‌ای وجود نداشت. می شه گفت فایده تجاری

نداشت که کسی بجای NT از لینوکس استفاده کنه. تا وقتی که آپاچی اومد و کلی چیزهای دیگه که به آپاچی وصل می‌شدن. می خوام بگم وقتی آدم می‌خواست بره و یسری سرور راه بندازه خیلی مقرون به صرفه‌تر بود. مقرون به صرفه‌تر از نظر پول واقعی که اون رو با لینوکس و آپاچی بسازه تا با IIS و NT . حتی اگر به این معنی باشه که آدم یه خرده پول خرج کنه. که به افرادش آموزش بده که چطور ازش استفاده کنن یا آدم‌هایی رو پیدا کنه که بهش واردن ولی خوبیش این بود که این دانش خیلی گرون نبود.چون کلی دانشجو وجود داشتن که مدت‌ها از لینوکس استفاده می‌کردن و خیلی باهاش آشنا بودن.

اریک ریموند:

اگه به منحنی استفاده وب سرورها نگاه کنید، آپاچی همش داشته سهم بیشتری از بازار رو بدست می‌آورده. یکسره داشته به رقیب‌های close sourceاش ضرر می‌زده و علتش اینه که مطمئن‌تره و انعطاف‌پذیرتره، گسترده‌پذیرتره. کارهایی رو که وب‌مسترها واقعا نیاز دارن رو می‌کنه. و ترکیب آپاچی و لینوکس راهش رو به تعداد بسیار زیادی از شرکت‌های تجاری باز کرده.

جایی که لینوکس ترکوند!!!!

BRIAN BEHLENDORF

آپاچی احتمالا روی لینوکس و FreeBSD بهتر اجرا می‌شه. و به همین دلیله که کامیونیتی‌های مربوط به این سیستم‌عامل‌ها هنوز بیشترین مشارکت رو در تولید آپاچی داشتن. و این که این‌ها سیستم‌عامل‌هایی بودن که تامین کنندگان خدمات اینترنت شدیدا شروع به استفاده از اون کردن. و تامین‌کنندگان خدمات اینترنت خیلی از آپاچی خوششون اومد چون بهشون اجازه می‌داد کلی کارهای متفاوت بکنن که بعضی از سرورهای تجاری

وب نمی‌کردن. مثل این قابلیت که هاست بیش از یک وب‌سایت روی یک دستگاه باشن که به وضوح اگر آدم یه ISP باشه و 40هزار کاربر داشته باشه و همشون بخوان برای خودشون وب‌سایت داشته باشن یه مقدار برای آدم مهم می‌شه.

یکی از عوامل کلیدی در رشد لینوکس، به وجود آمدن شرکت‌هایی بود که تخصصشان توزیع و پشتیبانی خود سیستم عامل بود. از میان این شرکت‌های نرم‌افزاری Red Hat از همه معروف‌تر است.

Red Hatیکی از توزیع‌های لینوکس بشمار می‌ره، داستان اون چی بوده؟

:DONNIE BRNES

Red Hat به عنوان یکی از محصول‌های مارک یوونیگ شروع شد. Marc Ewing وقتی که داشت توی IBM کار می‌کرد و توزیع لینوکسی می‌خواست که کمی بهتر باشه پس شروع کرد با لینوکس ور رفتن. بعد دید داره وقت بیشتری صرف نگهداری توزیع لینوکسش می‌کنه تا صرف پروژه جدیدش توی IBM. اینجوری شد که یجورایی خودش یه توزیع رو شروع کرد. بعد با باب یانگ Bob Young آشنا شد که اون موقع شرکتی به اسم کتاب‌فروشی ACC رو اداره می‌کرد. باب می‌خواست چیزی رو بفروشه که بیشتر مال خودش باشه تا محصولای آدمای دیگه رو. کار بازاریابیش خوب بود. مارک هم می‌دونست که توی بازاریابی به کمک احتیاج داره چون توی بحثای فنی خیلی وارد بود این شد که باهم متحد شدن.

ما در واقع توی آپارتمانی شروع به کار کردیم که مارک یووینگ توش زندگی می‌کرد. بعد یه روز دستشویی خونه ترکید و خونه همسایه پایینی رو آب برداشت. اونجا بود که ساکنان آپارتمان فهمیدن ما اونجا یه شرکت راه انداختیم و تصمیم گرفتن بندازنمون بیرون. یه هفته وقت داشتیم اولین دفترمون رو پیدا کنیم و کردیم.

نقش نت اسکیپ در مهم کردن لینوکس

اریک ریموند:

یه مقاله نوشتم به اسم کلیسا و بازار The Cathedral & the Bazaar و به این پرداخته بودم که اپن‌سورس چجوری کار می‌کنه. حتی اونموقع هنوز اسم "اپن‌سورس" رو نداشتیم و می‌گفتیم نرم‌افزار های آزاد. و گفته بودم که چرا ما تونسته بودیم نرم‌افزارهایی با کیفیت بی‌نهایت بالا رو تولید کنیم با اینکه مدام داشتیم قواعد مهندسی نرم‌افزار رو زیر پا می‌ذاشتیم.

فرق بین دو روش متفاوت برنامه‌سازی یا developeکردن برنامه رو توضیح می‌دادم. یکی روش سنتی بود یا همون closed development style که من اسمش رو گذاشته بودم روش کلیسایی. توی این روش آدم باید تک‌تک اهداف رو توضیح بده و گروه‌های پروژه کوچکی داشته باشه که با داشتن اختیارات سلسله مراتبی باهم کار می‌کنن و باید دوره انتشار محصولش هم طولانی باشه. از طرف دیگه اتفاقی که دیدم داره برای دنیای لینوکس می‌افته، روش خیلی فرد به فرد و نامتمرکز و شبیه یه بازار واقعی بود، که فواصل انتشار خیلی کوتاه بود و دائما از نظرات آدمایی که خارج از پروژه بودن بهره می‌برد.

بروس پرنز:

دلیل اهیمت نت اسکیپ Netscape’s اینه که اون‌ها اولین شرکت بزرگی بودن که به دنیای اپن سورس اومدن. البته ما سیگنوس رو برای ارائه پشتیبانی داشتیم ولی واقعا

چندان تجارتی در کار نبود. **نت اسکیپ اصلا برای این اپن‌سورس شد که یجوری با مایکروسافت بجنگه.** که داشت اینترنت اکسپلورر رو بیرون می‌داد. ولی اجازه نمی‌داد دیگران متن اون رو داشته باشن. اجازه نمی‌داد شرکت‌های دیگه همکاری کنن.

FRANK HECKER

(مهندس سابق نت اسکیپ)

من در بخش فروش کار می‌کردم. تصور خوبی داشتم که چرا مردم نرم‌افزارهای مارو می‌خرن و چه چیزهایی لازمه که نرم‌افزارهای ما در برابر محصولات رقیب در بازار موفق باشن. در هر حال مسئله این بود ما می‌دیدیم که با گذشت زمان نرم‌افزارهای ما با نرم‌افزارهای دیگه به خصوص مایکروسافت در رقابت بود. با گذشت زمان قیمت محصولات ما باید کمتر می‌شد چون بقیه داشتن محصولاتشون رو رایگان یا با بهای کم بیرون می‌دادن.

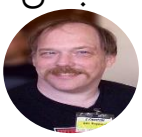
اریک ریموند:

اونا میترسیدن مایکروسافت بازار مرورگر رو به دست بگیره و از اون برای منحرف‌کردن استانداردهای HTTP و HTML که وب وابسته به اون‌هاست استفاده کنه و به محض

اینکه اون استانداردها رو به ابزار نگه‌داشتن کاربر تبدیل کردن بعدش می‌تونستن از اون تسلط استفاده کنن تا نت اسکیپ رو از بازار سرورها بیرون کنن. بازار سرورها هم جایی بود که نت اسکیپ ازش پول اصلیش رو در میاورد.

FRANK HECKER

آدم های کافی برای اینکه بتونیم نرم‌افزارهامون رو توی بازار زنده نگه داریم رو نداشتیم. مقاله کلیسا و بازار اریک ریموند نقش بزرگی برای انتشار سورس‌کد برنامه نت‌اسکیپ داشت. البته از قبلش داشتم به منتشر کردن سورس‌کدها فکر می‌کردم. یه مقاله نوشتم و توش ارجاع دادم به مقاله کلیسا و بازار. قصدمم این بود که یه بحث جنجالی بندازم توی نت اسکیپ که چرا باید سورس‌کد رو منتشر کنیم. اسم مقاله رو هم گذاشتم: «سورس‌کد برنامهٔ نت‌اسکیپ به عنوان محصولی از نت‌اسکیپ» عنوان می‌خواست اینو جا بندازه که سورس‌کد نه تنها ازش می‌تونیم در تهیهٔ محصول استفاده کنیم بلکه خودش هم یه محصول حساب می‌شه ! چیزی که مشتری‌ها و مردم دیگه بتونن ازش استفاده کنن. بعد فکر کردم خب لایسنسش رو چی بذاریم؟ چطور باید محصولاتمون رو توی این فضا بفروشیم؟ به رقبا نگاه کردم، مخصوصا مایکروسافت، اگر سورس‌کد رو منتشر کنیم اونا چه کارایی ممکن بود بکنن؟ آیا راهی بود که از همین سورس‌کدها علیه خودمون استفاده کنن؟



اریک ریموند:

از اینجا به بعد دیگه جریان اصلی توجه و اعتماد سرمایه‌گذاران رو به ما داد

FRANK HECKER

یه نسخه از مقاله رو دادم به مارک اندریسن Mark Andreessen یکی از بنیان‌گذاران نت اسکیپ اون زمان بود. اونم پخشش کرد و رسید به دست جیم بارکس دیل Jim Barksdale . بعد از مدتی هم اونا اعلام کردن که قصد دارن سورس‌کد نت اسکیپ رو

منتشر کنن! و اینجوری نت اسکیپ توجه عمومی رو به نرم‌افزارهای آزاد که بعدا گفتن اپن‌سورس رو جلب کرد. و سیستم عامل لینوکس در کانون توجه قرار گرفت.

لری آگوستین:

وقتی می‌گفتیم free مردم فکر می‌کردن مجانیه. فکر می‌کردن نمی‌تونن ازش پول دربیان یا اونو بفروشن که تصور اشتباهیه. می خواستیم این مفهوم رو برسونیم که نرم‌افزار آزاد یعنی اینکه کدش در دسترسه و بازه. یه جلسه گذاشتیم و به مفهوم اپن‌سورس رسیدیم. زنگ زدیم به لینوس گفتیم خوشت میاد ازش یا نه؟ براش جالب بود و خوشش اومد.

از طرفی هم ریچارد استالمن به طور رسمی اعلام کرد که با خونده شدن اپن‌سورس به جای نرم‌افزارهای آزاد کاملا موافقه. چون هدفش از آزادی این بود که مردم برای همکاری باهم و تشکیل کامیونیتی‌ها آزاد باشن و این برای کیفیت زندگی همه مهمه. و اومدن گفتن که اپن‌سورس 9تا حق مشخص رو به شما می‌ده:

بروس پرنز:

1-توزیع مجدد و آزاد

آزاد یا free ربطی به قیمت نداره. به معنی آزادیه. آدم باید آزاد باشه که بتونه نرم‌افزارش رو دوباره بین آدم‌های دیگه توزیع کنه. در واقع رایگان بودن یه تاثیر جانبیه. می شه بابت توزیع مجدد پول گرفت یا نگرفت.

2- در دسترس بودن سورس‌کد

باید سورس‌کد همراهش باشه که بشه برنامه رو نگهداری کرد. مثلا اگر بخوان از PC بجای Mac استفاده کنن باید بتونن نرم‌افزار رو تبدیل کنن.

3-کار اشتقاقی هم باید مجاز باشه

Derived Works have to be possible

اگر قرار باشه نرم‌افزار شما رو کسی گسترش بده باید قادر باشه که نتیجهٔ کار رو توزیع کنه. یه شرطی هم درباره یکپارچگی سورس‌کد مولف هست

4-یکپارچگی سورس‌کد مولف

می گه مولف می‌تونه یه جورایی نام خودش رو حفظ کنه. و اگر شما تغییری ایجاد کردید ممکنه مجبور باشید اسم برنامه رو تغییر بدید یا تغییراتتون رو خیلی واضح مشخص کنید که تغییراتتون به حساب مولف گذاشته نشه

5- نباید تبعیضی بین آدم‌ها و گروه‌ها باشه

6- نباید تبعیضی بین زمینه‌های کاری باشه

معنی‌اش اینه که از نرم‌افزار باید بشه چه در تجارت چه در مدارس استفاده کرد.

7- لایسنس باید قابل توزیع باشه

من باید قادر باشم که این لایسنس رو به شخص دیگه‌ای بدم و در این صورت لایسنس باید باز هم، اگه اون شخص اون رو به شخص سومی بده، کار کنه

8-نمی شه لایسنس مخصوص به یک محصول مشخص باشه

اگر من نرم‌افزارم رو روی سیستم ردهت توزیع کرده باشم لایسنس نمی‌تونه بگه که نمی‌شه این رو روی سیستم‌های SuSE یا Debian توزیع کرد.

9- لایسنس نمی‌تونه نرم‌افزارهای دیگه رو محدود کنه

اگر من این رو روی یک CD با نرم‌افزار دیگه‌ای منتشر کنم، لایسنس نمی‌تونه بگه اون برنامهٔ دیگه هم باید آزاد باشه وگرنه نمی‌شه برنامهٔ من رو توزیع کنین

اریک ریموند:

یه اتفاق دیگه این بود که پایگاه‌های داده به این جمع اضافه شدن! اواخر ژوئیه و اوایل اوت بود که اوراکل اعلام کرد برنامه‌هاشو به لینوکس حمل می‌کنه . و همینطور Sybase و دیگر سازندگان مهم پایگاه داده این رو اعلام کردن.

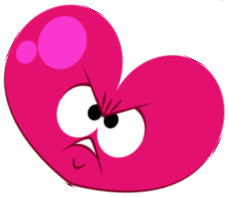
لری آگوستین:

رفته رفته لیست مخاطب‌ها و مشتریان اپن‌سورس زیاد شدو کسایی مثل سیسکو ، شرکت دات کام داشتن وارد این لیست می‌شدن. بالاخره توجه سرمایه‌گذارها جلب شد. عکس لینوس روی مجله فورچون چاپ شد و این یعنی توی اپن‌سورس داشت اتفاقایی می‌افتاد. دیگه سرمایه‌گذارا نمی‌تونستن نادیده بگیرن و از دست من که مدام بهشون سر زده بودم خسته شده بودن. آخرش به این نتیجه رسیدن که سرمایه گذاری کنن.

اریک ریموند:

مایکروسافت داره از لینوکس برای دفاع از خودش استفاده می‌کنه. اون ها از لینوکس برای اثبات این ادعا استفاده کردند که اون‌ها انحصار رو در دست ندارن چون اصولا لینوکس می‌تونه هر لحظه اون‌ها رو از موضع برتری که دارن پایین بکشه. بحث خیلی جالبی بود. یه ادعای کاملا غلط انداز، چون به هیچ وجه پاسخی به این اتهامات نداد که اون‌ها قبلا

مشغول قلدر بازی و دیگر کارهای غیررقابتی بودن. ولی اون‌ها زرنگی کردن و البته در این مورد، قاضی زیر بار نرفت.



درباره میلیاردها دلار ثروتی که از ساختهٔ تو دراومده و چیزیش مستقیما به تو نرسیده چه احساسی داری؟

لینوس توروالدز:

اگه من لینوکس رو به همه نمی‌دادم اونجوری هم هیچ پولی بهم نمی‌رسید. یعنی می‌خوام بگم این یه موقعیت بدون باخته. این مسئله که کلی شرکت‌های تجاری وجود دارن به این معنیه که کلی آدم مرتبط با لینوکس هست که قبلا وقت اضافی‌شون رو روی لینوکس کار می‌کردن و حالا برای کاری که دوست داشتن بکنن پول می‌گیرن. این مسئله از این جهت به من کمک می‌کنه که من به هرحال می‌خواستم اون‌ها روی لینوکس کار کنن.

ریچارد استالمن:

کل پروژه گنو در واقع یک هک بزرگه. کاری بزرگ از سر هوشمندی شورشی و بازی‌گوشانه برای بهترکردن جامعه. من فقط به دنبال بهتر کردن هستم. ولی به روش هوشمندانه

شما لینوکسی؟

حالا هر چی، امر تون؟؟!!!

چی شد که شما پنگوئن‌ها شدین نماد لینوکس؟ مثلا این مینیون‌ها چشون بود؟ یا فیل‌شاه که کارتونشم دارین. اصلا یه سوال کلی‌تر، چرا هر چی جک و جونوره شده نماد زبان‌برنامه‌نویسی؟ چرا یه هیولارو نمی‌ذارن؟ ما به این گوگولی‌مگولی!!!!

اییییییییییش. نخیرم. یه روز لینوس مارو توی باغ‌وحش دید و عاشقمون شد و یه نوع پنگوئن به اسم tux رو که چاق و خوشحاله رو گذاشت روی لینوکسش

Oh my GOD

چه خفن، خیلی تعجب کردم تو قیافمم معلومه؟

چقد پنگوئن خوشگلی شما!!! لینوس حق داشته ها!

پسر امروز چند شنبه اس؟

هشت‌شنبه است دیگه. چطور؟

عارضم خدمتتون که ما توی هشت‌شنبه‌ها مخ نمی‌زنیم

اونوقت بر چه اساس؟

بر اساس اصل خودم. اصل خرها.

دلیل از این بالاتر؟

نگاه کمی فنی‌تر به سیستم‌عامل

همان‌طور که لینوس‌توروالدز در ابتدای داستان اشاره کرد، لینوکس یک سیستم‌عامل کامپیوتری است. یک سیستم‌عامل شامل نرم‌افزارهایی است که کامپیوتر شما را مدیریت می‌کند و اجازه می‌دهد برنامه‌های خود را روی آن اجرا کنید. و چندین ویژگی دارد که به آن اشاره می‌شود:

شناسایی و تنظیم سخت‌افزارها: وقتی که لینوکس بوت می‌شود (یعنی وقتی کامپیوترتون رو روشن می‌کنین)، به اجزاء سیستم شما نگاهی می‌اندازد (مثل CPU، کارت شبکه و...) و بعد نرم‌افزارها (درایورها و ماژول‌ها) و مواردی که نیاز به دسترسی به این سخت‌افزارهای خاص را دارند لود می‌کند.

سخت‌افزار در واقع هر چیز فیزیکی است که بتوانیم به آن دست بزنیم، مثل رم، هارد دیسک و...

مدیریت پروسه‌ها یا فرآیندها: سیستم‌عامل باید همزمان چندین کار را اجرا کند و از طرفی تصمیم بگیرد که کدام باید به CPU دسترسی پیدا کند و چه زمانی باید این دسترسی صورت بگیرد. هم‌چنین سیستم‌عامل باید راه‌های برای شروع، متوقف‌کردن، و تغییر وضعیت فرآیندها و پروسه‌ها را ارائه دهد.

مدیریت حافظه: رم و بخش swap (جلوتر می‌گیم چیه) باید به برنامه‌هایی که نیاز به حافظه دارند، داده شود. سیستم‌عامل این تصمیم را می‌گیرد که چگونه به حافظه یا همان موری درخواست بدهد و آن‌را مدیریت کند.

ارائه‌دادن یک رابط کاربری: یک سیستم‌عامل باید راهی برای دسترسی به سیستم را برای یوزر ایجاد کند. سیستم‌هایی که از لینوکس استفاده می‌کنند، چیزی دارند به اسم shell . در واقع برای کار با سیستم لینوکس، ما درون shell برنامه یا دستورات را می‌نویسم و اجرا می‌کنیم. اشتباه نکنید! برای اینکه از لینوکس استفاده کنید لزوما به این معنی نیست که باید از shell استفاده کنید. این روزها با آمدن توزیع‌های مختلفی، مثل ubuntu، رابط های گرافیکی دارند که می‌توان بدون داشتن دانش استفاده از shell و دستورات یا به

اصطلاح کامندلاین‌ها، از لینوکس استفاده کرد. درست مثل چیزی که در ویندوز شاهد آنیم! یعنی ویندوز را نصب می‌کنید و وارد یک محیط گرافیکی می‌شوید و کارهایتان را انجام می‌دهید. ولی قدرت استفاده از لینوکس، همین داشتن دانش استفاده از shell و کامندلاین‌هاست. البته بیشتر مبتدی‌ها از این بخش فرار می‌کنند و به لینوکس پشت می‌کنند. اما مطمئن باشید اگر مفهومی آن را متوجه شوید، استفاده از آن قدرت شیرینی به شما می‌دهد.

کنترل فایل سیستم‌ها: در واقع سیستم‌عامل، مالکیت و دسترسی به فایل‌ها و دایرکتورها (فولدرها)ی موجود را کنترل می‌کند.

ایجاد دسترسی برای کاربران و احراز هویت آن‌ها (شناسایی آن‌ها): ایجاد یوزر اکانت‌ها و تعیین محدوده برای کاربران. کاربران و گروه‌های کاربری را از هم جدا می‌کند تا بتوانند هرکس به فایل‌های مربوط به خودش دسترسی داشته باشد و یا فرآیندها و پروسهٔ خودشان را انجام دهند و کنترل کنند.

ارائه امکانات مدیریتی: در لینوکس شاید صدها یا هزاران دستور و پنجره‌های گرافیکی وجود دارد تا کارهایی هم‌چون اضافه‌کردن یوزرها، مدیریت دیسک‌ها، مدیریت شبکه، نصب نرم‌افزار، و به طور کلی ایجاد امنیت و مدیریت کامپیوترتان را فراهم کنند.

راه‌اندازی سرویس‌ها: برای استفاده از چاپگرها، رسیدگی به پیام‌های ورودی، و ارائه انواع خدمات مربوط به شبکه و... باید در سیستم‌عامل انجام شود. سرویس‌های زیادی در لینوکس اجرا می‌شود.

لینوکس راه‌های زیادی برای اجرا و متوقف‌کردن این سرویس‌ها را در اختیار می‌گذارد.

ابزارهای برنامه‌نویسی: طیف وسیعی از ابزارهای برنامه‌نویسی و لایبرری‌های مختلف برای ساختن نرم‌افزارها و اپلیکیشن‌ها موجود است.

به عنوان کسی که می‌خواهد سیستم‌های لینوکسی را مدیریت کند، باید بدانید که هر بخش چگونه کار می‌کند. با اینکه بسیاری از کارها را می‌شود با استفاده از رابط گرافیکی انجام داد

ولی فهم و دانستن دستورات یا همان کامندلاین‌ها در shell برای کسی که مدیریت سیستم‌های لینوکسی را دارد یک "باید" به شمار می‌رود.

ویژگی‌های پیشرفته لینوکس که اغلب در شرکت‌های بزرگ استفاده می‌شود، به شکل زیر است:

کارهای پیشرفته لینوکس، همچون clustering (خوشه‌بندی)، virtualization (مجازی‌سازی)، cloud computing (پردازش یا رایانش ابری)، Real-time computing (یا رایانش بی‌درنگ! برای همینه که توی صفحه اصلی، گفتم از ترجمه‌کردن اصطلاحاتی که مال ما نیستند خوشم نمی‌آید!)، specialized storage (ذخیره‌سازی تخصصی) بیشتر در شرکت‌ها استفاده می‌شود.

چه فرقی بین لینوکس و سیستم‌عامل‌های دیگه هست یا؟

اگر تازه پا به دنیای لینوکس گذاشتید، حتما گذرتون به ویندوز مایکروسافت یا مک افتاده است. اما فرق اساسی که با لینوکس دارند در زیر به آن‌ها اشاره شده است:

نمی‌توانید کدهایی که برای ایجاد سیستم‌عامل استفاده شده است را ببینید،

بنابراین نمی‌توانید سیستم را مطابق نیازهای خودتان و انتظاراتی که از آن دارید تغییر دهید.

نمی‌توانید کدها را برای پیداکردن باگ‌ها چک کنید. راحت بفهمید که ایراد کار از کجاست یا اگر مشکل امنیتی دارد از کجا آب می‌خورد. و از همه مهم‌تر نمی‌توانید یاد بگیرید که یک کد چگونه کار می‌کند.

ممکن است به راحتی نتوانید نرم‌افزارهای خودتان را به سیستم‌عامل وصل کنید تنها به این دلیل که سازندگان سیستم‌عامل شما می‌خواهد این انحصار را حفظ کند!

شاید پیش خودتان بگویید چه اهمیتی داره؟ من که نمی‌خوام دولوپر نرم‌افزار بشم. نمی‌خوامم ببینم که چطوری می‌شه سیستم‌عاملم رو تغییر بدم.

شاید حق با شما باشد. اما حقیقت این است که بسیاری از افراد و شرکت‌های دیگر که پیشگام خلاقیت و نوآوری هستند از این امکانات و نرم‌افزارهای اپن‌سورس و بعضا رایگان استفاده می‌کنند، مثلا گوگل، فیسبوک و...

از طرفی لایسنس ویندوز رایگان نیست و اکثر ما به طور غیرقانونی در حال استفاده از آن هستیم. مثلا هنگام نصب ویندوز کافی است قبل از زدن تیک گزینه License Agreement حوصله به خرج دهید و متن آن‌را بخوانید! به طور وحشتناکی اشاره می‌کند به کلی از کارهایی که نباید بکنیم و می‌کنیم.

در بخشی از بندهای آن اشاره شده: لایسنس این نرم‌افزار به شما فروخته شده است نه خود نرم‌افزار. یا مثلا در جایی دیگر اشاره شده است که فقط می‌توانید خودتان از این نرم‌افزار استفاده کنید آن‌هم تنها بر روی یک کامپیوتر، نباید به کسی آن را بدهید و...

اگر جزء اقیلیتی هستید که لایسنس را پرداخت کردید، مطمئنا هم در این دنیا و هم در آن دنیا رستگارید! (فقط دست مارم سر پل بگیرید!)

خب که چی؟ می‌خوای بگی تو الان لایسنس شو داری؟ و مثلا کیلی کیلی فارسی بلد نیستی؟

در کل نمی‌خواهم بگویم استفاده از مایکروسافت بد است و اخ! هنوز هم که هنوز است بسیاری از نرم‌افزارها هم‌چون فتوشاپ، یا سیویل‌تری‌دی و اتوکد و... در محیط لینوکس وجود ندارد. البته مشابه آن وجود دارد. مثلا بجای وردآفیس، چیزی هست به اسم LibreOffice؛ یا بجای فتوشاپ، Gimp وجود دارد ولی به هیچ عنوان به کاملی فتوشاپ و یا نرم‌افزارهای اصلی در ویندوز نیست و تنها پاسخ‌گوی بخشی از نیازهای کاربران است.

شاید ورود به دنیای لینوکس کمک کند تا بیشتر از اینی که هست مصرف‌گرا نباشیم. و چیزهای جدید را تجربه کنیم.

از طرفی چون شرکت‌ها و کمپانی‌های بزرگی از لینوکس استفاده می‌کنند، پس به افراد بیشتری نیاز دارند که این مهارت را داشته باشند که با سیستم‌های لینوکسی کار کنند و آن‌ها را مدیریت کنند.

توزیع لینوکس چیه؟

اگر داستان را خوانده باشید، فهمیدید که لینوکس مستقیما اشاره به کرنلی دارد که لینوس تورووالدز ساخت و هر سیستمی که از این کرنل استفاده کند، اصطلاحا به آن گنو/لینوکس می‌گویند.

حالا کرنل را لینوس‌تورووالدز ساخت و نرم‌افزارها و یا بهتر بگوییم پکیج‌ها را هم گنو تولید کرده بود. پس به زبان ساده اگر این‌ها را کنار هم بگذاریم و کامپایل کنیم، یک سیستم‌عامل خواهیم داشت. یعنی اینجوری هر کسی توانایی این را دارد که یک لینوکس برای خودش بسازد. به این لینوکس‌هایی که هر کسی برای خودش مطابق نیازش می‌سازد، "توزیع" یا "distribution" می‌نامند. (ولی عامیانه بهش می‌گن distro)

بعضی از این distroها را شرکت‌ها ساختند. یعنی پشت یک distro، یک شرکت سفت و سخت نشسته و مدام در حال آپدیت‌کردن پیکیج‌ها و سایر موارد آن distro است. وقتی بحث شرکتی پشت یک distro می‌آید، یک کلمه Enterprise به آن می‌چسبد. (مثلا جلوتر در مورد distroای که شرکت رد هت یا Red Hat ساخت حرف می‌زنیم، اونجا هم همین کلمه به لینوکسش چسبیده!)

شرکت‌های مطرحی هم‌چون Red Hat و Debian و... به این ترتیب لینوکس خودشان را از پایه ساختند. در حال حاضر کلی distro داریم.

در واقعیت ساختن یک سیستم‌عامل کار چندان آسانی نیست، چون جدا از اینکه باید برای کامپایل‌کردن و کنار هم قراردادن پکیج‌ها تلاش کنیم، بحث پشتیبانی آن هم هست. یعنی اگر برنامه‌ای آپدیت شد باید تست کنیم و بعد ارائه کنیم و کلی کارهای دیگر.

حالا ینی چی؟ راهی نداره؟ ما می‌میریییییم ما می‌میرم

چرا، بسیاری از شرکت‌ها همین distroهایی که وجود داشت، مثل Debian (دبیان)، کمی تغییر دادند و distro خودشان را ارائه کردند. (اینم یکی از مزیت‌های دنیای اپن سورسه که می‌تونی هر چیزی رو مطابق نیازت تغییرش بدی) در واقع با اینکار بعضی از distroهای محبوب، پایهٔ شکل‌گیری و تولد distroهای دیگر شدند. مثلا distroمحبوب ubuntu بر پایهٔ Debian ساخته شده است.

در این بین برخی distroها به کل تولدشان برای مصارف خاص و مشخصی بود. مثل: KNOPPIX، Gentoo، Mandrake که بعدها به اسم Mandriva تغییر کرد.

نمی‌خواهم پیاز داغ ماجرا را زیاد کنم، اما شکل‌گیری لینوکس تنها یک سیستم‌عامل نبود و نیست. اگر داستان را دنبال کرده باشید، متوجه شدید که پشت آن فلسفه‌ای وجود دارد. آن هم فلسفه آزادی.

لینوکس اگر رایگانه پس ملت پول و درآمدشون رو از کجا در میارن؟!

این‌جاست که باید تفاوت بین دو واژه free با freedom مطرح شود. همان‌طور که در بندهای تعریف اپن‌سورس گفته شده، درست است که لینوکس اپن‌سورس می‌باشد اما لزوما به معنای رایگان بودن نیست. و این یعنی سورس‌کدها در دسترس هست اما ممکن است خود آن "پولی" باشد. یعنی در این بین ممکن است خود فروش سورس‌کد هم یک محصول به حساب بیاید. مثل کاری که شرکت Red Hat کرد. علاوه بر این شرکت‌ها بابت خدماتی که ارائه می‌دهند هم هزینه می‌گیرند.

از Red Hat تا Debian

خب این اسمای کج و کوله چیه؟

من می‌دونم! اسم دوتا شرکت یا بالاخره چاییه که از اپن‌سورس و دنیای لینوکس شروع کردن پول دراوردن و خدمات دادن

Red Hat در اواخر سال 1990 به چند دلیل محبوب شد:

یکی استفاده از package managment های RPM بود.

دلیل دیگر محبوب‌شدن این distro، آسانی نصب‌اش بود. یکسری سوالات ساده را پاسخ می‌دهید که در بیشتر اوقات هم گزینه‌های دیفالت دارد.

دلیل دیگرش استفاده از محیط گرافیکی بود. با این کار یوزرها دیگر نیازی نداشتند که کامند یا دستوری را یاد بگیرند تا بتوانند از لینوکس استفاده کنند. در محیط گرافیکی ابزارهایی گذاشته شد که یوزر می‌توانست تاریخ و زمان، و کارهای پایه و اولیه خودش را انجام دهد.

سال‌های زیادی بود که Red Hat، distroمحبوبی برای هر دو قشر شد. هم برای کسانی‌که حرفه‌ای کار می‌کردند و هم برای کسانی که علاقمند به دنیای لینوکس بودند. به تدریج که دنیای لینوکس گسترده می‌شد و افراد بیشتری وارد آن می‌شدند باعث شد مرز این جنبش و فضا به جایی فراتر از Red Hat بروند. Red Hatهم کار قبلی را رها کرد و بجای آن شروع کرد و دو سیستم‌عامل را توسعه داد: Red Hat Enterprise و Fedora.

سال 2012 این شرکت اولین شرکت اپن‌سورسی بود که به درآمد سالانه بیش از 1میلیارد دلار رسید. این درآمد را از طریق فروش محصولاتی در حوزه Red Hat Enterprise Linux یا به اختصار RHEL بدست آورد.

با این حال، distroهای دیگر تمرکز خودشان را روی دسکتاپ‌ها (کامپیوترهای شخصی) یا مشاغل کوچک گذاشته بودند. اما Red Hat بیشتر روی ویژگی‌ها و برنامه‌هایی کار می‌کرد که برای کسب‌وکارها و دولت نیاز بود.

Fedora

با اینکه Red Hat، RHEL را عرضه کرده بود، اما distroای به نام Fedora را ارائه کرد برای کسانی که می‌خواستند از لینوکس برای مصارف شخصی و رایگان استفاده کنند و از طرفی بتواند این شرکت را در کامیونیتی دوست‌داران لینوکس نگه دارد.

Fedora بیش از 16 هزار پکیج نرم‌افزاری را دارد. بسیاری از این‌ها با آخرین تکنولوژی اپن‌سورس در دسترس هستند. به عنوان یک کاربر یا یوزر، شما قادرید از آخرین نسخه آن برای server (سرور) یا دسکتاپ و کارهای دیگر به رایگان استفاده کنید.

چون Fedora تمرکزش بیشتر بر این است با آخرین تکنولوژی پیش برود، پس خیلی با ثبات نیست! بنابراین برای اجرای بعضی از نرم‌افزارها یا انجام یکسری کارها، باید کارهای اضافی‌تری هم برای به راه‌افتادن کارتون انجام بدید.

با این‌حال نباید به آسانی از این توزیع گذشت، Red Hat بسیاری از اپلیکیشن‌های جدید را در Fedora تست می‌کند. با کارکردن با این disrto می‌توانید مهارت‌هایی که برای کار با بعضی ویژگی‌های Red Hat Enterprise Linux هست را بدست آورید.

با اینکه Fedora محبوب است اما در سال‌های اخیر distroای به نام ubuntu توانسته بیش از Fedora محبوب شود.

Ubuntu بر پایهٔ distro محبوب Debian ساخته شده است. و نسبت به Fedora از ثبات بیشتری برخوردار است. بسیاری از distroهای دیگر بر پایهٔ Debian ساخته شده، که می‌توانید در سایت distrowatch.com پیدایشان کنید. بیش از 130 توزیع فعال وجود دارند که برپایهٔ Debian هستند. بعضی از distroهای محبوب برپایهٔ Debian این‌ها هستند: Linux Mint، elementary OS (من برای اولین بار که وارد دنیای لینوکس شدم از این توزیع استفاده کردم!)، Zorin Os ، LXLE، Kali Linux (بیشتر برای امنیت بکار می‌ره) و بسیاری از توزیع‌های دیگر. ولی محبوبیت بیشتر Debian بخاطر پیدایش Ubuntu (ubuntu) بود.

نصب این distro بسیار ساده است. از محیط گرافیکی خوبی استفاده می‌کند و برای نصب، برخی نرم‌افزارها را پیشنهاد می‌دهد. (یه چیزی مثل بازار برای اندورید یا play store داره که نرم‌افزارهایی رو برای دانلود گذاشته)

حتی می‌توانید ubuntu را رویCD یا فلش به صورت لایو load کنید و از آن استفاده کنید. لینوکس لایو به لینوکس‌هایی گفته می‌شود که کلا از روی سی‌دی می‌تواند بوت شود و همه کارهایتان را می‌توانید انجام دهید. و اگر هم بخواهید چیزی را save کنید به یک storageدیگه نیاز دارید. بیشتر به این درد می‌خورد که اگر یک سخت‌افزاری دارید که دچار

مشکل شده است اینطوری تست کنید، یا می‌خواهید به هر دلیلی سیستم‌عامل خودتان را جابه‌جا کنید.

برای شروع چی بزنیم؟

یعنی چی؟ خانواده اینجا نشسته‌ها! الان قهر می کنه می‌ره دیگه ادامه‌شو نمی گه. شما به دل نگیر. منظور اینه از چه توزیعی استفاده کنیم؟ اونم برای شروع؟

وقتی پا به دنیای لینوکس می‌گذارید، اولین سوالی که پیش می‌آید این است که با چه distro یا توزیعی کار را شروع کنم؟

وقتی می‌خواهید از ویندوز استفاده کنید، یکی از ورژن‌ها (مثلا 10 یا 7 و...) را انتخاب می‌کنید و نصب می‌کنید. ویندوز مشخصا توسط یک کمپانی (مایکروسافت) تولید می‌شود اما لینوکس اینگونه نیست. و مردم روی بخش‌های مختلف این سیستم‌عامل کار می‌کنند.

ما دنیایی از distroها را داریم، اما چیزی که مطرح است این است که باید بدانیم که گاهی می‌خواهیم از لینوکس روی دسکتاپ کامپیوتر (کامپیوترای شخصی) استفاده کنیم و گاهی روی سرور. مثلا ممکن است آن‌هایی که برای سرور هستند رابط گرافیکی نداشته باشند.

در واقع 70 درصد توزیع‌ها مشابه یکدیگرند و در بعضی بخش‌ها مثل مدیریت بسته یا پیکیج منیجمت فرق دارند. همان‌طور که ریچارد استالمن اشاره کرده، سیستم‌عامل‌های یونیکسی (مثل گنو/لینوکس) چیزی جز بسته‌ها نیستند. حالا فرق بعضی از distroها در مدیریت این بسته‌هاست. مثلا Red Hat از RPM استفاده می‌کند ولی distroهایی که

مبتنی بر debian(دبیان) مثل ubuntu، از مدیریت بسته از dpkg استفاده می‌کنند. برای همین خیلی درگیر انتخاب distro نشید، برای شروع یکی را انتخاب کنید و بعدها می‌توانید عوض کنید.

برای همین ما در این کتاب از ubuntu شروع می‌کنیم. پیش از این هم به دلایل‌اش اشاره شده. با این حال اگر از distroهای دیگر استفاده می‌کنید مشکلی با بحث‌ها نخواهید داشت.

با لینوکس چی کاره می‌شم؟ مصلا زن و بچم می‌تونن جوجه‌ای که شما آدما می‌خورین رو بخورن؟

بیا فک و فامیلتو جمع کن ببر آبرومون رو ببرن ای بابا. بلد نیستی سوالشم درست بنویسه. مصلا آخه یا مثلا؟

شما به گندگی خودت ببخش. هیولان. ندیدن دیگه.

بسیاری از مشاغل مهندسی، شاید لینوکس را در کنار کارهای دیگرشان و برای اجرای ایده‌هایشان استفاده می‌کنند. با این حال، استفاده از لینوکس در دنیای امروزی با شدت زیادی رو به افزونی است. پس زمینه‌های کاری زیاد هم به طبع به وجود می‌آید. با این حال، مشاغلی که به طور مستقیم به لینوکس اشاره دارند:

مهندس نرم‌افزار، پشتیبانی سیستم، برنامه‌نویسی روی لینوکس، آموزش، مهندس شبکه، متخصص سرور، کار با LAMP و...

نکته: اگر مشخصا و یک‌راست! می‌خواهید آزمون‌های Lpic را شرکت کنید، بسیاری از افراد در سطح اینترنت به رایگان در حال آموزش این آزمون هستند. مثلا جادی‌عزیز.

روند پیشرفت این کتاب به سمت دست و پنجه نرم‌کردن با لینوکس! و همینطور استفاده از آن در دنیا و ... خواهد رفت. با این حال برخی مطالب مربوط به آزمون‌ها گفته خواهد شد.

(اینو گفتم برای کسایی که صاف میخوان برن فقط آزمون lpic رو بدن که این کتاب یه cook book نیست که بتونید اون آزمون رو بدید، منابع زیادی توی نت هست که بیشتر به کمکتون میاد.)

حوصلم سر رفت. می گی اصل کاری رو یا نه؟

نصب لینوکس

چون distroهای مختلف توسط شرکت‌های مختلف ساخته شده‌اند، پس مراحل نصب هم ممکن است تفاوت داشته باشد. (مسئله شاید خیلی جدی نباشه! هر distroای که دستتون بیاد و دوست داشته باشید نصب کنید، کافیه یه سرچ کنید که چجوری نصبش کنم.)

ولی چیزهایی که بین نصب distroها مشترک است و البته مهم، یکی نحوه پارتیشن‌بندی و دیگری انتخاب فایل سیستمی است که می‌خواهید انتخاب کنید. والبته بحث نرم‌افزارهایی که می‌خواهید داشته باشید. (یا همون نصب پکیج‌ها)

(گفتیم لینوکس در واقع اشاره مستقیم داره به کرنلی که لینوس ساخت و هر کی این کرنل رو استفاده کنه اصطلاحا بهش میگن لینوکس داره! پس تا الان فهمیدیم که لینوکس با distro فرق داره! پس وقتی می‌گه لینوکس، یعنی هر سیستمی که از این کرنل استفاده می‌کنه نه لزوما یک distro مشخص)

اگر نمی‌خواهید به طور کامل از ویندوز به لینوکس مهاجرت کنید و قصد دارید دو موتوره از سیستم‌تون لذت ببرید، سه راه وجود دارد:

1- درون ویندوز برنامه‌ای برای مجازی‌سازی مثل virtual box نصب کنید و لینوکس را درون آن بریزید
2- می‌توانید از دو سیستم عامل روی یک کامپیوتر بهره ببرید: کلا بعد از بوت‌کردن سیستم شروع به نصب‌اش کنید(یعنی روی خود سیستم‌تون نصب کنید) در راه حل دوم هر بار وقتی که سیستم را روشن کنید از شما می‌پرسد که آماده‌اید با ویندوز پرواز کنید یا لینوکس! در واقع می‌توان لینوکس را به صورت dual boot در کنار ویندوز نصب کرد. فقط اینکه شما می‌توانید تنها یک boot loader داشته باشید. (روی سکتور صفر فقط یکی میره) چون بوت لودر ویندوز نمی‌تواند بوت لودر لینوکس را بالا بیاورد پس بهتر است اول ویندوز را نصب کرده و بعد اقدام به نصب لینوکس کنید.
3- اگر از ویندوز 10 استفاده می‌کنید، آپ‌های لینوکس را از store ویندوز دریافت کنید و محیط کامندلاین آن‌را بررسی کنید.

لطف می کنی پیش غذا رو بگی؛ موارد لازم جهت طرز تهیه‌اش. انگار کلاس آموزش لینوکس یا مانع هستیم

اگر می‌خواهید روی virtal box یا هر ماشین مجازی دیگری نصب کنید، اول آن را دانلود کنید.

نسخه آخر ubuntu را هم اینجا می‌توانید پیدا کنید. (اگر نشد دستی سرچ کن: ubuntu.com اون بالا بگرد یه گزینه download پیدا می‌کنی)

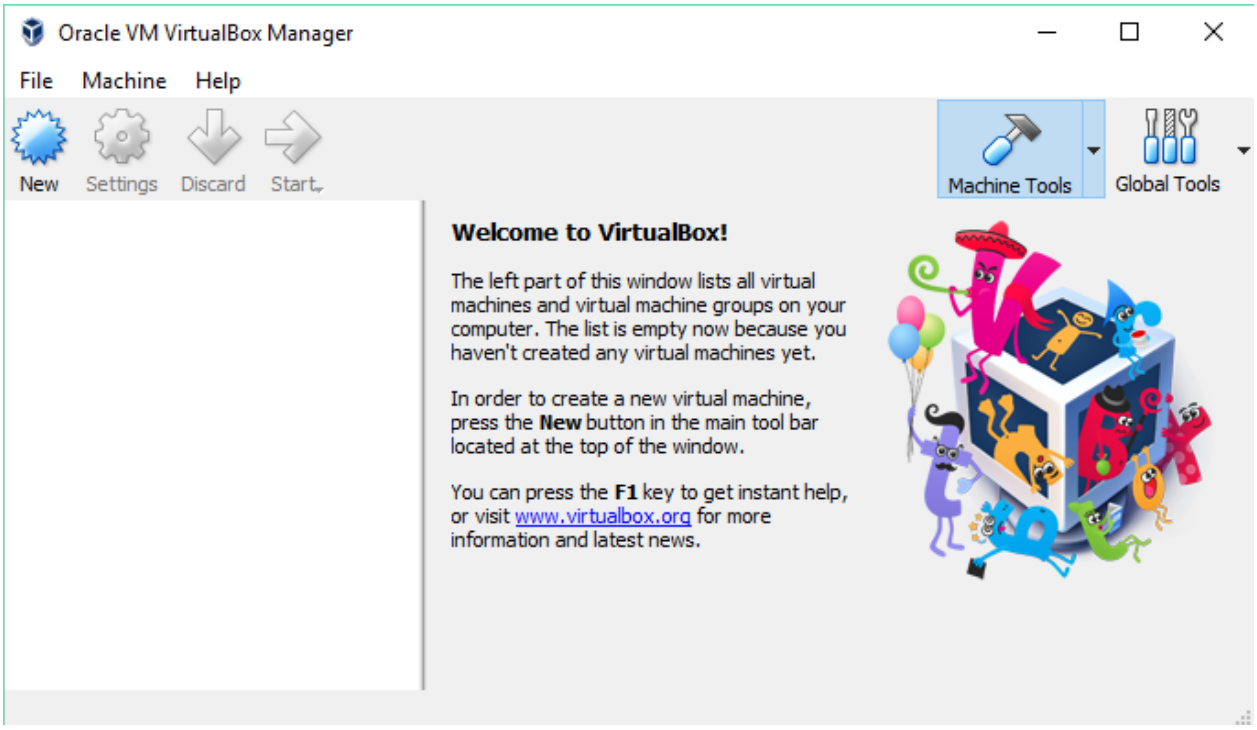
اها من پیدا کردم یه فایله iso اومد برای دانلود

باشه تو خوبی عزیزم

حدود هفت الی هشت گیگابایت هم دست کم نیاز دارید.

نصب لینوکس روی virtual box

Vitrual box را باز کنید، روی new کلیک کنید



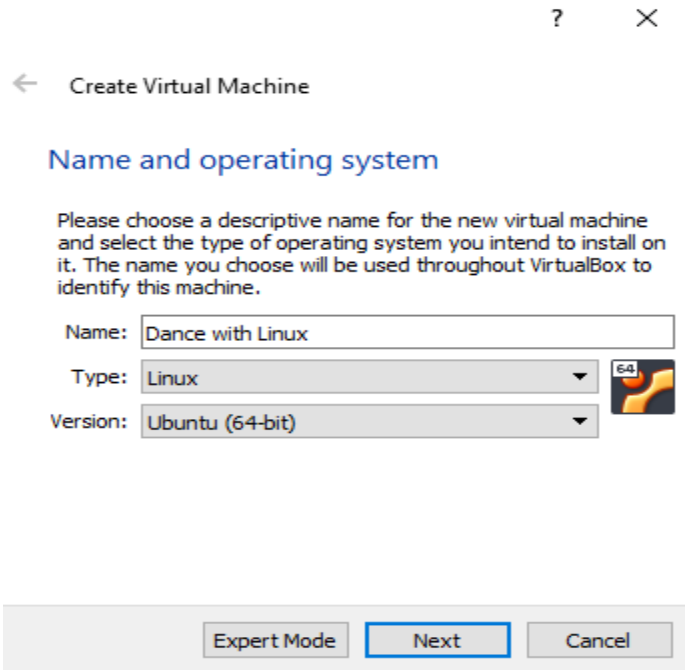
در قسمت Name اسم انتخاب کنید.

من دوست دارم اسمشو بذارم دلبر دلبرم، تاج سرم،

فکرت نمی‌ره از سرم....

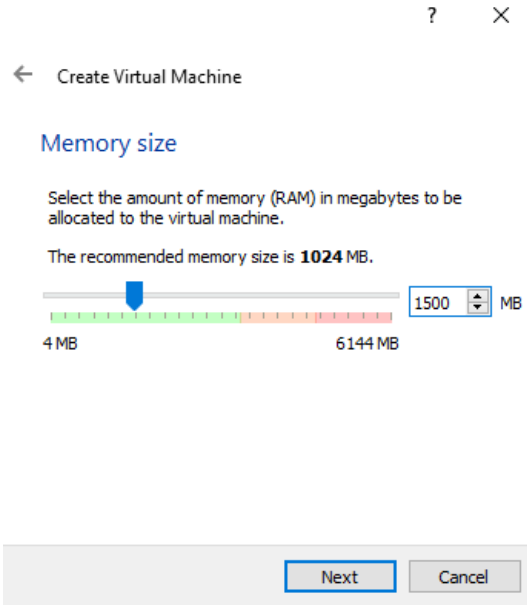
پسه کَرمون کردی. نخون

در قسمت type، linux را انتخاب کنید. در قسمت versionهم مطابق چیزی که دارید

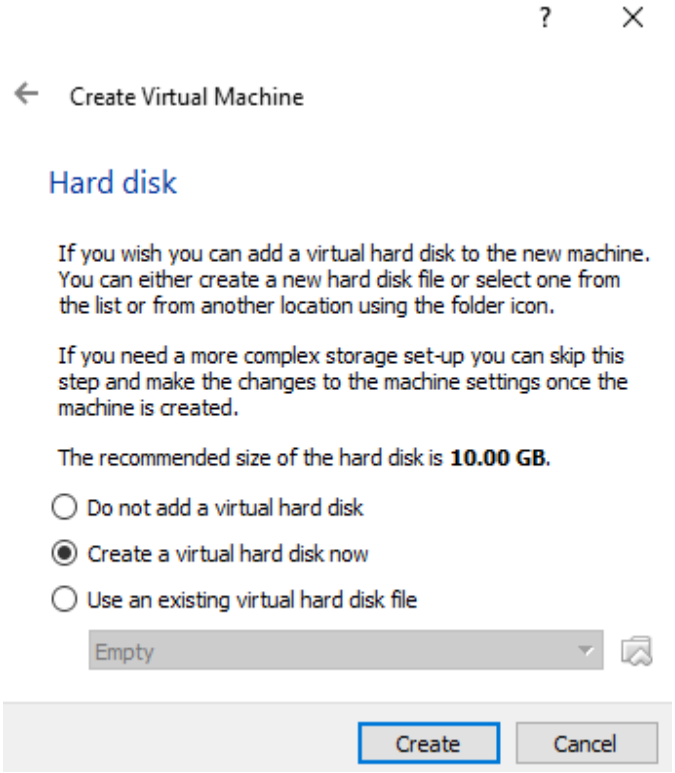


اگر هم distroشما در گزینه‌ها نبود، در گزینه‌های ابتدایی لیست، آخرین کرنل را انتخاب کنید.

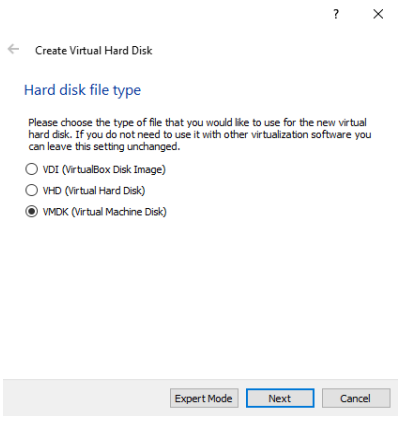
در قسمت Memory size میزان RAMای که می‌خواهید را انتخاب کنید.



در قسمت Hard disk روی create a virtual hard disk now قرار بدهید و گزینه create را بزنید.



اگر روی گزینه VMDK بگذارید، می‌توانید این از آن در دیگر سیستم‌هایی که ماشین مجازی دارند نیز استفاده کنید.



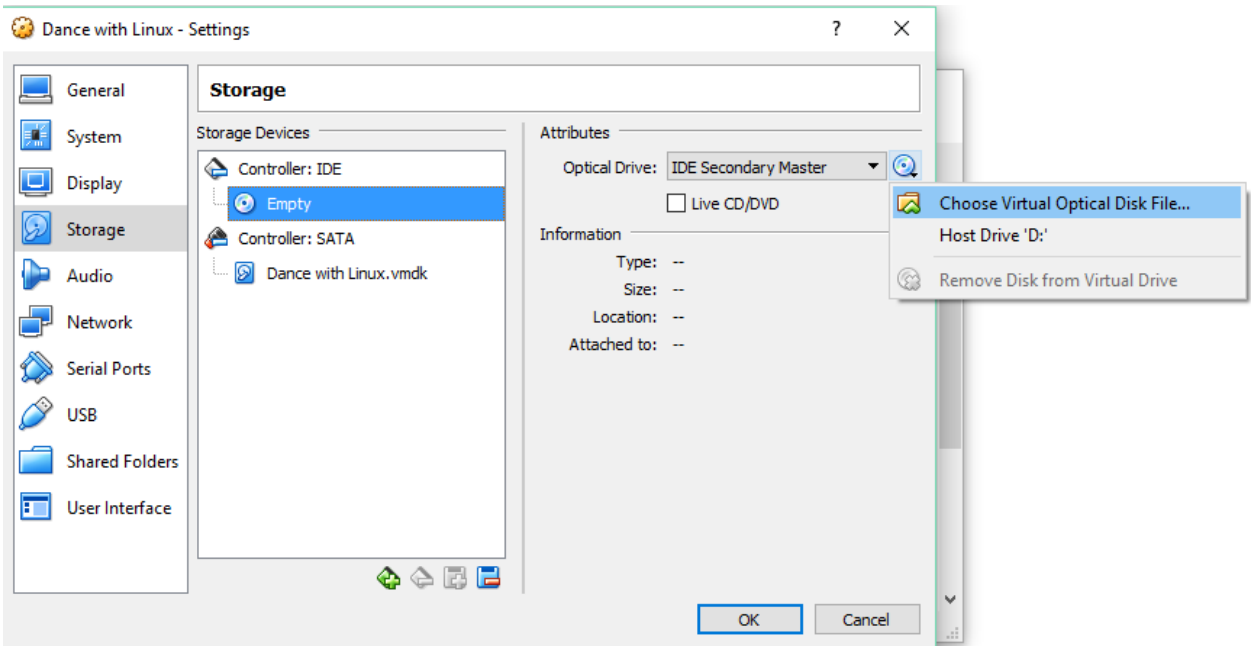
در قسمت storage on physical hard disk روی گزینهٔ dynamically بگذارید.

در قسمت بعدی یک اسم برای hard disk بگذارید. (یا همان اسم قبلی را بگذارید.)

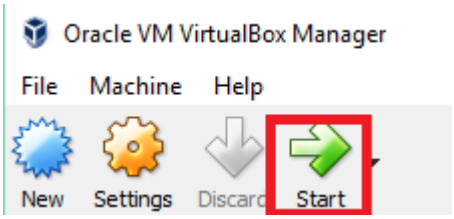
تا الان پیش زمینه آماده شد. یعنی ماشین مجازی شما آماده است. حالا باید ubuntu را نصب کنید.

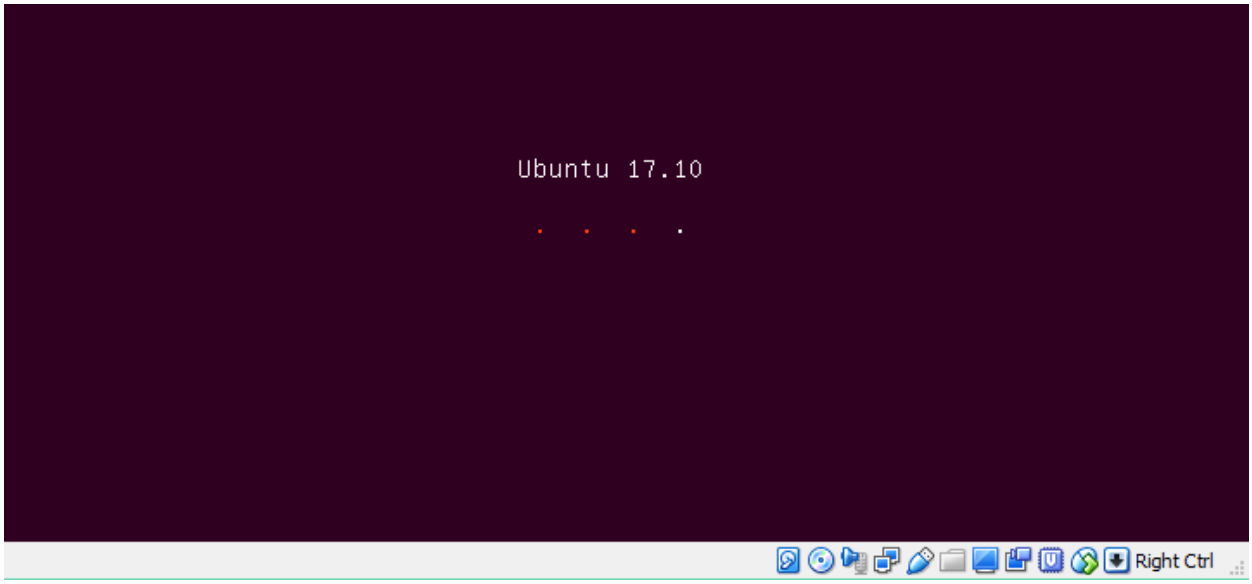
روی setting کلیک کنید. (بالا دومین آیکن)

در بخش storage روی Empty بزنید و در سمت راست یک آیکن CD می‌بینید، کلیک کنید و فایل iso که مربوط به فایل ubuntu دانلود شده است انتخاب کنید.

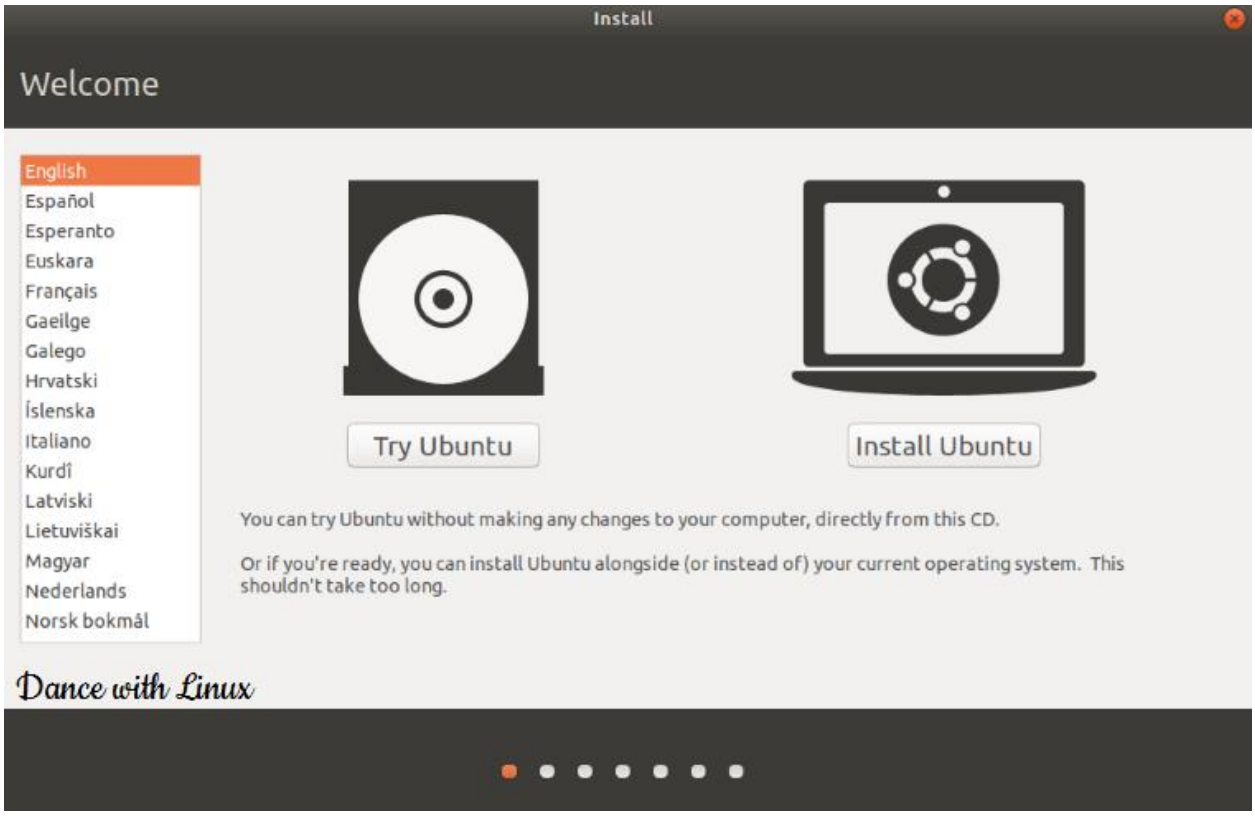


بعد روی continue و در آخر روی start بزنید.

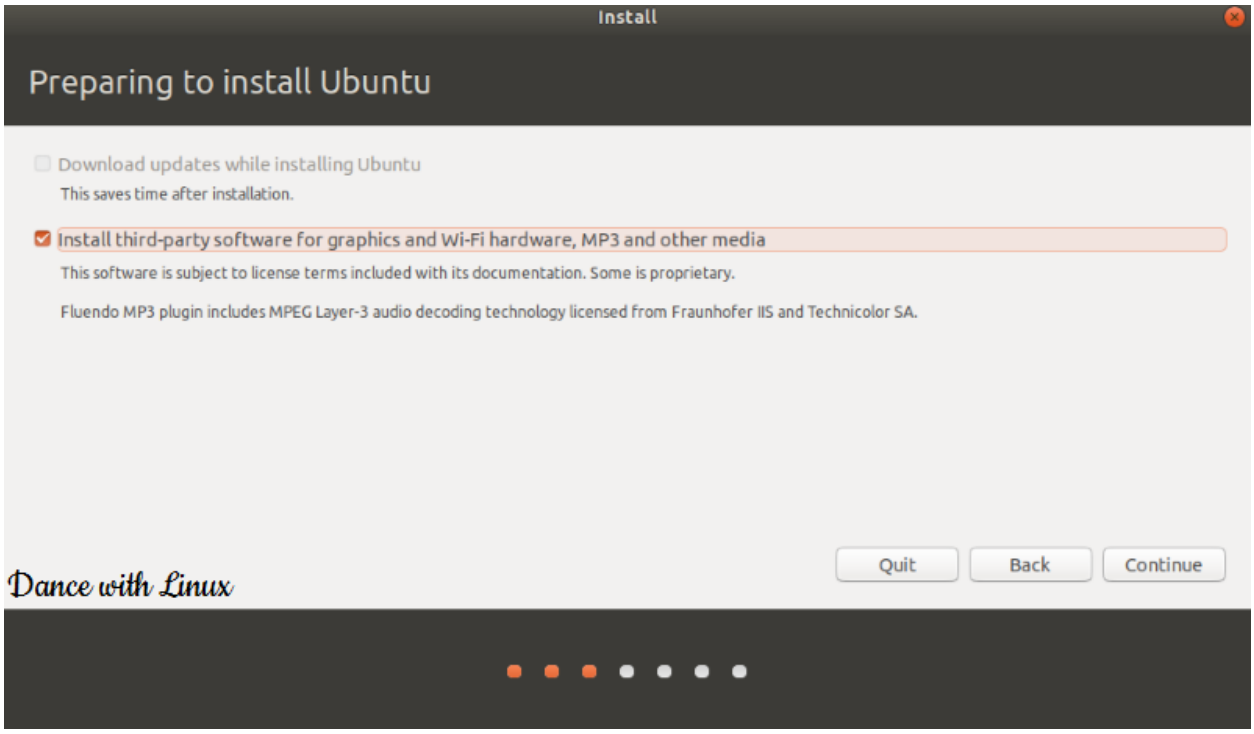




روی گزینه ubuntu install کلیک کنید.



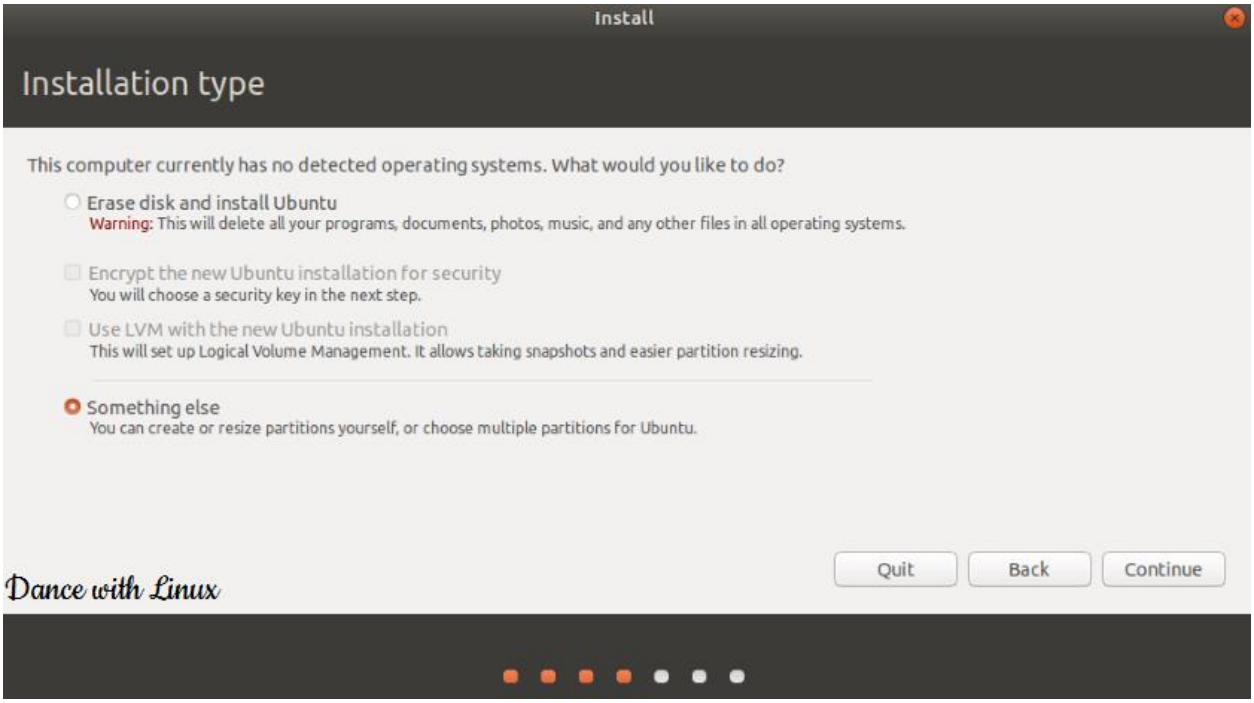
در قدم بعدی اگر تمایل دارید آپدیت‌ها را موقع نصب دانلود کنید، سیستم‌تون را به اینترنت وصل کنید و تیک گزینه download updates while installing را بزنید.



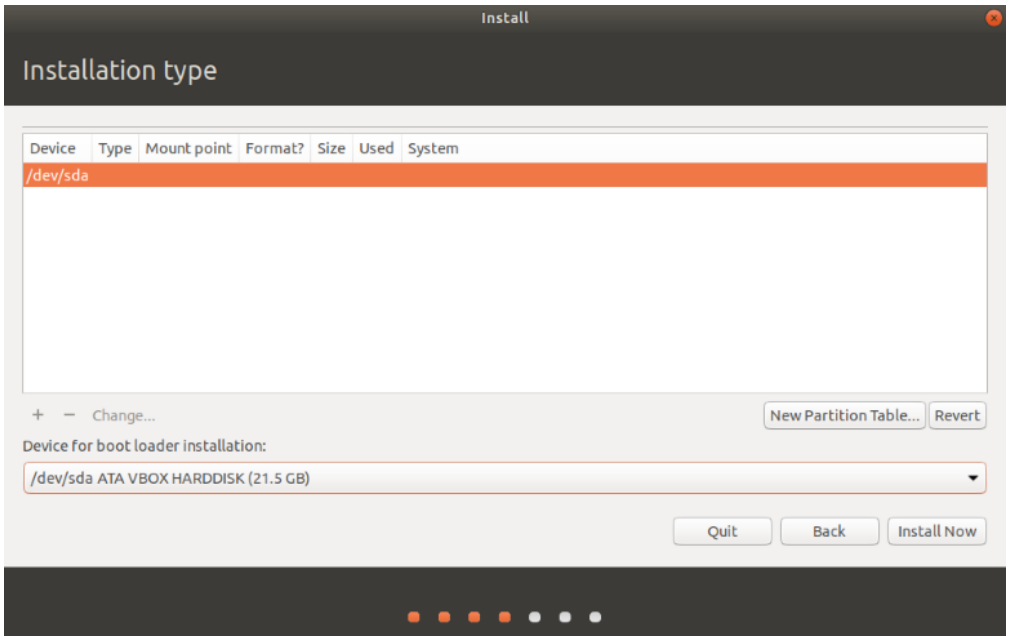
اینجا بخشی است که انتخاب می‌کنیم از چه پارتیشن‌بندی‌ای می‌خواهیم استفاده کنیم و چه فایل سیستمی مد نظر ماست.

در گزینه اول، که دیفالت نصب هم است، به طور کامل هارد دیسک شما را پاک می‌کند و پارتیشن‌های مورد نیاز را نصب می‌کند. این گزینه بیشتر به درد کسانی می‌خورد که تازه وارد فضای لینوکس شده‌اند. با زدن این گزینه همه کار را خود نصب‌کننده انجام می‌دهد. منتهی برای توضیح برخی نکات ما گزینه sth else را زدیم.

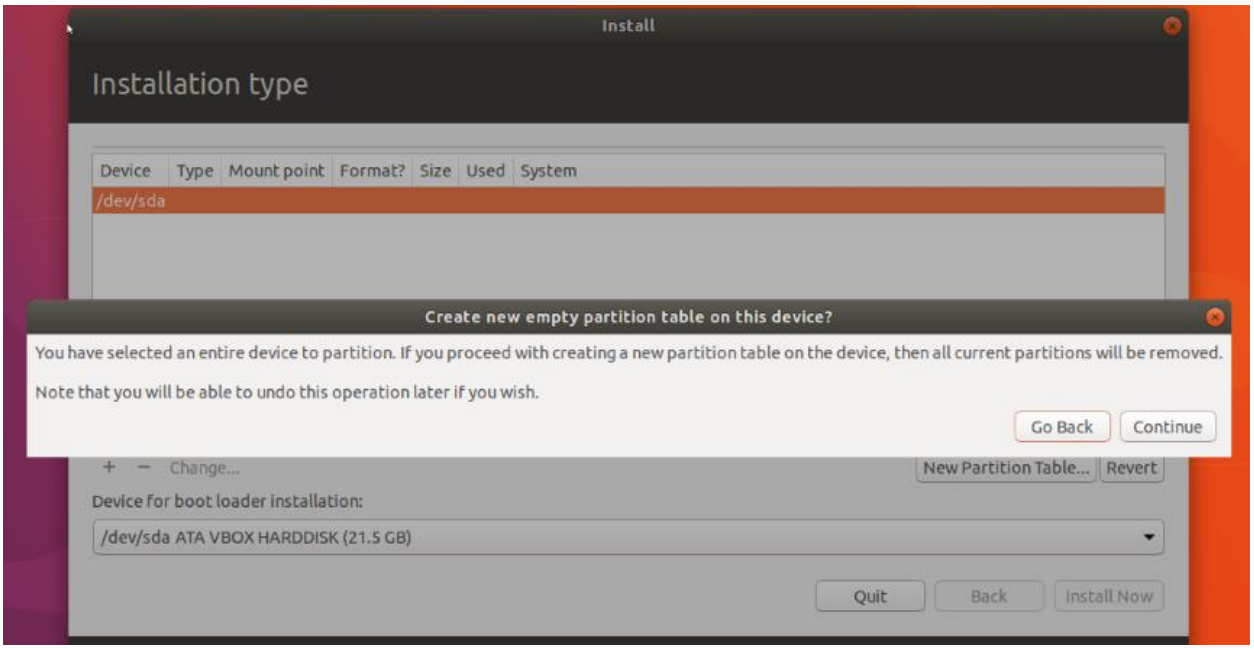
هشدار: اگر بصورت dual boot لینوکس را نصب کنید، با زدن گزینه اول، همه هارد دیسک شما را پاک می‌کند.



قسمت بعدی مربوط به لیست هارد دیسک‌های شماست. همه چیز در لینوکس به صورت فایل به شما ارائه می‌شود. dev در حقیقت فولدری است که سخت‌افزارها را به صورت فایل نشان می‌دهد و sda/ هم مربوط به اسم هارد دیسک شماست.

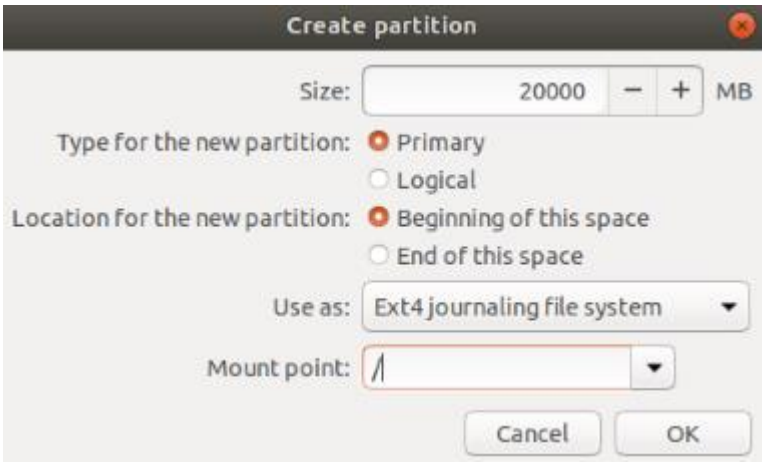


این هارد دیسک الان پارتیشن ندارد. روی new partition بزنید و بعد روی کانتینیو.



برای ساخت پارتیشن روی + پایین سمت چپ بزنید. اندازه پارتیشن را انتخاب کنید. نوع دیسک را روی primary و جای آن را روی اول دیسک یعنی beginning of this space بزنید. گزینه بعدی اشاره دارد که وقتی پارتیشن شما ساخته شد با چه فایل سیستمی فرمت شود. (بگذارید همان دیفالت یعنی Ext4 jornaling file system بماند.)

در قسمت Mount point یک اسلش یعنی / بذارید با این کار می‌فهمانید که می‌خواهید لینوکس شما در این پارتیشن نصب شود. اگر نگذارید، صرفا یک پارتیشن ایجاد شده است که لینوکس روی آن نصب نمی‌شود و نمی‌توانید به مرحلهٔ بعد بروید.



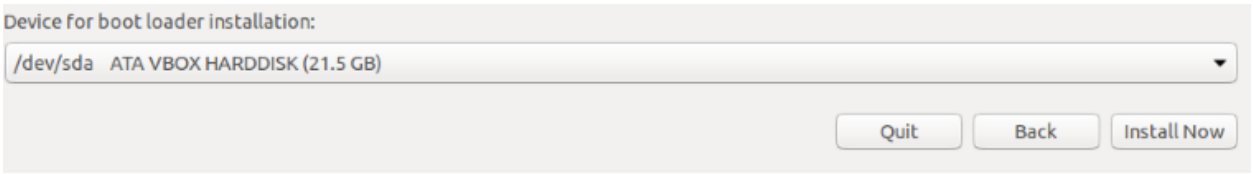
اگر فضای خالی داشته باشید می‌توانید یک پارتیشن دیگر نیز بسازید. بهتر است یک پارتیشن برای swap بسازید.

Swap چیه؟ اینْ دومینْ پاره داره می گیاااا. اگه فحشّه بگو صدا بزْنم هیولاهای (بچه‌ها) بالا بیاناااا

در یک سیستم‌عامل وقتی رَم پر می‌شود، سیستم دیگر قادر به اجرای پردازش‌های جدید نیست. بنابراین نیاز به یک موری خالی داریم. برای راهکار، اگر ما یک پارتیشن را به swap اختصاص بدهیم، لینوکس قادر خواهد بود که بخش اشغال شده موری را به swap بفرستد تا جا برای اجرای پردازش جدید باز شود. توجه داشته باشید که از swap نمی‌توان به عنوان موری اصلی استفاده کرد فقط بخشی را می‌توان به آن فرستاد و بعدا باید آن را برگردانیم به سرجایش. اگر سیستمی کارش به swap بیافتد بسیار کند خواهد شد و این هشداری است برای اینکه بدانید باید فکری به حال حافظه یا افزایشش بکنید.

برای ساختن این پارتیشن، این بار روی logical بزنید، و در قسمت use as گزینهٔ swap area را بزنید.

در قسمت بعدی باید انتخاب کنیم که boot loader روی کدام پارتیشن نصب شود. در واقع boot loader لینوکس را load می‌کند. گزینه را روی هارد دیسک بگذارید. . (دلیل فنی: نیاز دارید boot loader روی سکتور صفر قرار بگیرد.)



در آخر روی گزینه install و بعد continue بزنید.

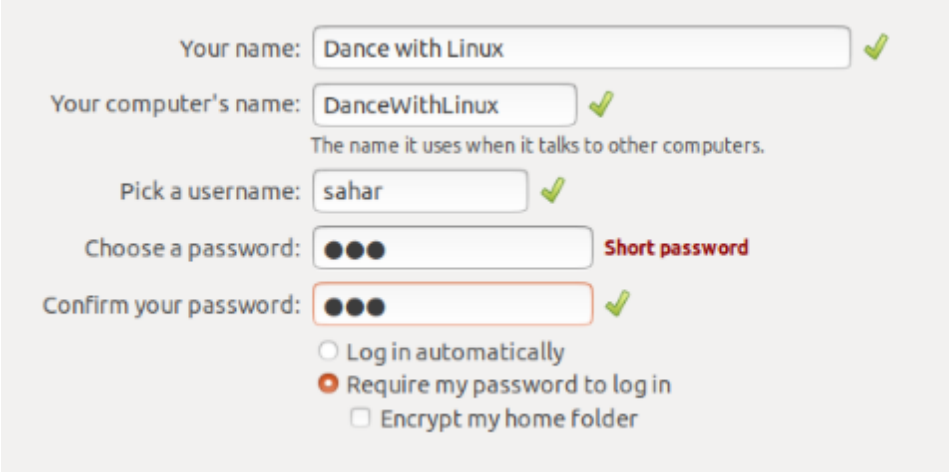
کشور را انتخاب کنید.

توصیه: کیبور را روی انگلیسی بگذارید. بعدا می‌توانید کیبور فارسی را اضافه کنید.

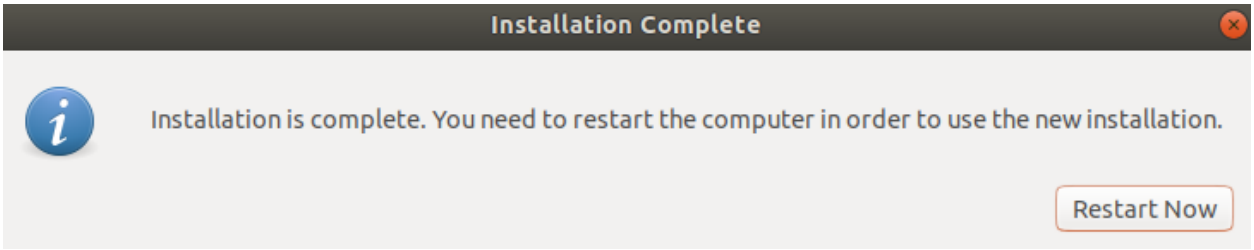
حالا باید یک یوزر بسازید.

یک پسورد انتخاب کنید.

توصیه: پسورد ساده‌ای انتخاب کنید چون بارها اتفاق می‌افتد که موقع اجرای کامندی، باید پسورد خود را وارد کنید. با انتخاب گزینهٔ log in automatically موقع ورود به سیستم از شما پسورد نمی‌خواهد اما با انتخاب گزینهٔ بعدی، از شما واردکردن پسورد را می‌خواهد.



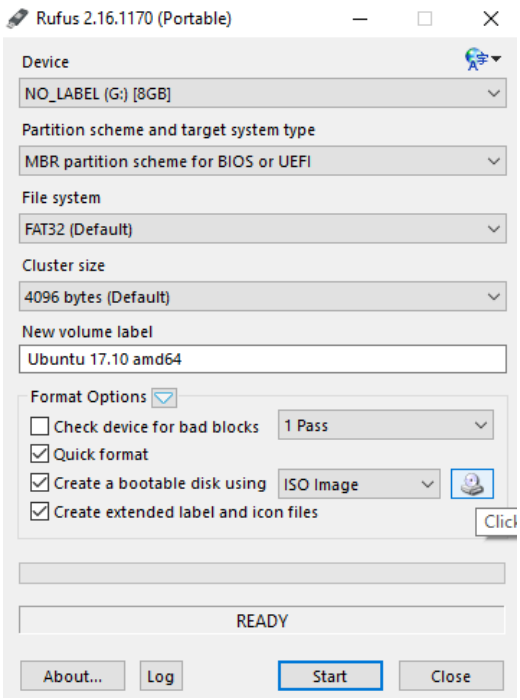
در انتها هم پس از نصب پیغام restart سیستم را می‌بینید.



نصب لینوکس روی سیستم

برای نصب لینوکس روی سیستم، بیش از هر چیز، نیاز دارید یک فلش یا DVDای را بوتیبل کنید. برای این کار بسیاری از نرم‌افزارها وجود دارند. مثلا ما از rufus استفاده کرده‌ایم.

بعد از دانلود نرم‌افزار آن را باز کنید.



در قسمت Device فلشی که برای بوتیبل‌کردن انتخاب کردید را انتخاب کنید!

عین خیابانی حرف می‌زنی! انتخاب رو انتخاب می‌کنن دیگه.

برای نحوه پارتیشن‌بندی هم، MBR را انتخاب کنید و برای فایل سیستم هم FAT32. در انتها فایل iso دانلود ubuntu را از آیکن جلویی iso Image انتخاب کنید و در انتها start. در ادامه هشدار می‌دهد که تمام فایل‌ها فلش شما پاک خواهد شد. دوتا اوکی بدین و تموم.

بعد سیستم را ریستارت کنید و فلش را بوت کنید. (یعنی موقعی که صفحه مشکی اومد جلوتون یکی از کلیدارو بزنید می‌ره و فلشتون رو بوت می‌کنه. اگر نشد، یعنی در قسمت تنظیمات بوت شما، گزینه این که اول فلش را بوت کنه فعال نیست، اونو فعالش کنین.)

بعد از بوت کردن فلش، مراحل نصب به ترتیبی است که در تصاویر بالا، از قسمت install آمده است.

فصل دوم: قلب لینوکس

معرفی کرنل

کار با shell

شروع کار با دسکتاپ

دسکتاپ یکی از شخصی‌ترین بخش‌های زندگی یک آدم به شمار می‌رود! نحوه چینش آیکون‌ها و اینکه چه فایل‌ها و برنامه‌هایی روی صفحه باشند یا حتی اینکه از چه دسکتاپی استفاده کنیم، همگی تابع سلیقه آدم‌هاست بیشتر.

شنیدی که چی گفت؟ گفت دسکتاپ شخصی‌ترین بخش کامپیوتره.

مثل مسواج

دلبندم مسواک

پس من عکسای شخصی آینده خودم و لینوکس‌جان و بچه‌هامون رو می‌ذارم روی دسکتاپ. ژست‌های این شکلی هم می‌گیرم

خری دیگه خر

پس چی. من افتخار می‌کنم به خر بودنم. اصلا هر کی بجز من بگه خرم، باهاش قانونی مطابق قانون خودم برخورد می‌شه.

چون لینوکس یک سیستم چندکاربره یا multiuser به حساب می‌آید برای ورود به سیستم اولین کاری که می‌کنید لاگین کردن است.

وقتی لاگین می کنم دقیقا چی می‌شه؟

سوالو نگاه. وقتی به خونتون می‌رسی دقیقا به چی دسترسی پیدا می کنی؟

هر چی با من بحث نکن. فرق داره.

چه فرقی؟

وقتی از بیرون میام خونه یا همین لاگین می کنم اولین چیزی که بهش دسترسی دارم سرویس بهداشتیه.

ولی وقتی خونه باشم و لاگین کنم به خونه، اولین چیز میتونه خیلی چیزا باشه. بحث فلسفیه. از درک تو خارجه. باید توی موقعیتش باشی تا بفهمی. اصلا بیا بریم خونمون بهت بگم.

الان که فکر می کنم می‌بینم فلسفت خیلی سنگینه. همینجا می‌مونم ولی همین که جای دستشویی گفتی سرویس بهداشتی جای شکرش باقیه. این فلسفتم نگه دار به لینوکس بگو و ببرتش خونت نه من. شعور می‌خواد که از تو کلا خارجه.

جواب بچه رو بده

چشم تو فقط عصبی نباش. وقتی لاگین می کنی تقریبا همین وضعیته که خره کفت. یعنی وقتی لاگین می کنی می تونی به shell و دسکتاپ مختص به خودت و کلی اجازه‌های دیگه دسترسی داشته باشی.

چون لینوکس یک سیستم چندکاربره یا multiuser به حساب می‌آید برای ورود به سیستم اولین کاری که می‌کنید لاگین کردن است. شاید به طور خلاصه بتوان گفت که عمل loginکردن سه کار اصلی را انجام می‌دهد:

1- شما را به عنوان یک کاربر خاص شناسایی کرده
2- Shell یا پوسته و دسکتاپ مختص به خودتان را ارائه می‌دهد
3- مجوزهایی را برای دسترسی به فایل‌ها و یا تغییر آن‌ها را ارائه می‌دهد. چون هر کاربری متناسب با اینکه چه نوع کاربری است، سطح دسترسی متفاوتی دارند.

می‌خوام دستشویی رو از سطح دسترسیت خارج کنم تا فلسفمو بفهمی

عجب خریه‌ها، کی اصلا همچین دسترسی رو به تو میده؟

هرچی! اختیار خونه خودمو که دارم

خونت نمیام

پس عروسیمم نیا. چون اونجا دیگه خیلی چیزارو از سطح دسترسیت خارج می کنم. مثلا اون دختره... آره همون، با شرک آشناش می کنم... یجوری از سطح دسترسی خارجت می کنم که همش بوق اشغال بزنی

چرا کلا بحث لاگین را مطرح کردیم؟ چون در لینوکس می‌توان چندین سطح دسترسی تعریف کرد اما قصد اصلی این بوده که بتوانیم تفاوت بین یوزر عادی یا regular user را با کاربر روت یا root user را بیان کنیم. (به کاربر روت، کاربر ریشه هم گفته می‌شود.)

کاربر عادی: شخصی است که از سیستم استفاده می‌کند. یک نام کاربری منحصر به فرد و کلمهٔ عبور مختص به خودش را دارد. یک سیستم می‌تواند چندین کاربر عادی داشته باشد. به طور کلی، کاربر عادی تنها اجازه استفاده از سیستم را دارد.

کاربر root: هر سیستم لینوکس، فقط یک کاربر root دارد. این کاربر اجازهٔ دسترسی‌های خاص را دارد و اگر به عنوان کاربر root لاگین کنید باید خیلی مواظب باشید چون اشتباهات در این ناحیه صدمات جدی دارد. به کاربر root اصطلاحا super user نیز گفته می‌شود.

این کاربر کنترل کامل یک سیستم‌عامل در اختیارش است. توانایی اینکه هر برنامه‌ای را باز کند یا اجرا کند یا پاک‌کردن یوزرهای دیگر را نیز دارد.

چرا باید بینمون فرق وجود داشته باشه؟ این فاصله طبقاتی توی لینوکس چیه که راه انداختین؟ لابد فردا حس نژاد پرستی تون گل می‌کنه و ما رو هم پیرون می‌کنید. اصلا اینو جواب بده: از یه کامپیوتر که فقط یه نفر استفاده می‌کنه چرا اکانت‌ها متفاوت باید وجود داشته باشه؟

برای اینکه کاربر عادی که هر روز از لینوکس استفاده می‌کند اشتباها فایل‌های حیاتی سیستم را تغییر ندهد. در واقع این کار بیشتر برای آسایش است! در غیر این صورت یک اشتباه می‌تواند به طور کلی سیستم را به باد فنا دهد. به همین دلیل بجز کاربرعادی یک کاربر root هم دارید.

جدا از این این بحث کاربرها این امکان را فراهم می‌کند که اگر چندین نفر از یک سیستم لینوکس استفاده می‌کنند، بتوانند کاربرها در صورت تمایل، فایل‌هایشان را از دسترس قرار گرفتن یا تغییر داده‌شدن توسط کاربرهای دیگر محافظت کنند.

شبکه بندی احتمالا بهترین دلیل برای بکارگیری یک سیستم لینوکسی می‌باشد. این داشتن نام کاربری یا اکانت مجزا و منحصر به فرد چندین فایده می‌تواند داشته باشد. مثلا اکانت شما می‌تواند با سایر منابع کامپیوترها در ارتباط باشد، مثل ارتباط داشتن با سیستم فایل‌ها، نرم‌افزارها، mailboxها و...

در اکثر مواقع، پیش می‌آید که شما اکانت خودتان را به اصطلاح شخصی‌سازی می‌کنید. برای مثال ممکن است برای اسامی فایل‌ها، نام‌های استعاری به کار ببرید. یا برنامه‌هایی

تنظیم کنید. با جمع‌آوری این اطلاعات در یک نقطه، انتقال‌دادن اکانتتان یا اضافه‌کردن یک اکانت جدید به یک کامپیوتر دیگر در آینده آسان می‌شود و سردرگم نخواهید شد.

نمی‌خوای ما رو برگردونی توی فضای دسکتاپ و یه گشتی بزنیم؟

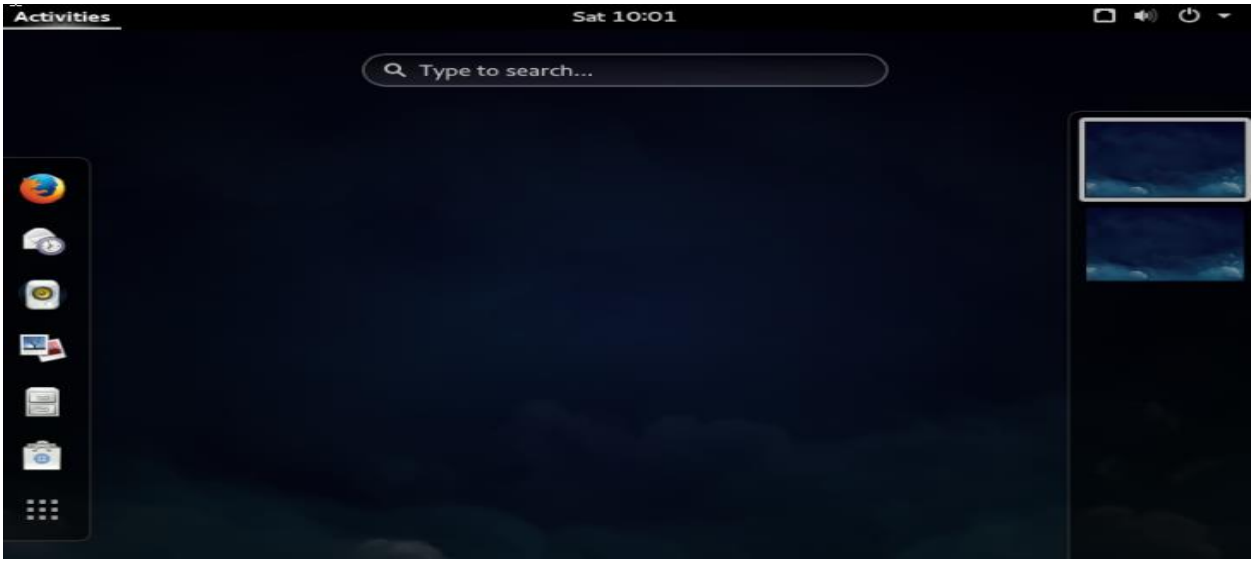
یارونم میاد؛ چه فضا عاشقانه می‌شد. من باشم، لینوکس باشه. توی حیاط دسکتاپ راه بریم...پارک هم بود... البته گشت ارشاد مال ما نیست مال آدماس.

قدم‌زدن در فضای دسکتاپ

محیط گرافیکی یا GUI در واقع همان چیزی است که وقتی Desktopبالا می‌آید می‌بینید. وقتی بحث محیط های گرافیکی می‌شود، برخلاف ویندوز که یک محیط گرافیکی واحد دارد، لینوکس این چنین نیست. در لینوکس‌های مختلف، این محیط‌های گرافیکی متفاوت است اما در ویندوز این چنین نیست. مثلا اگر از ویندوز 10 استفاده می‌کنید محیط گرافیکی شما با تمام کسانی که از ویندوز 10 استفاده می‌کنند یکی است. اما دسکتاپ‌ها در دنیای لینوکس بسیار متنوع‌اند و این حق انتخاب برای تمامی کاربران وجود دارد.

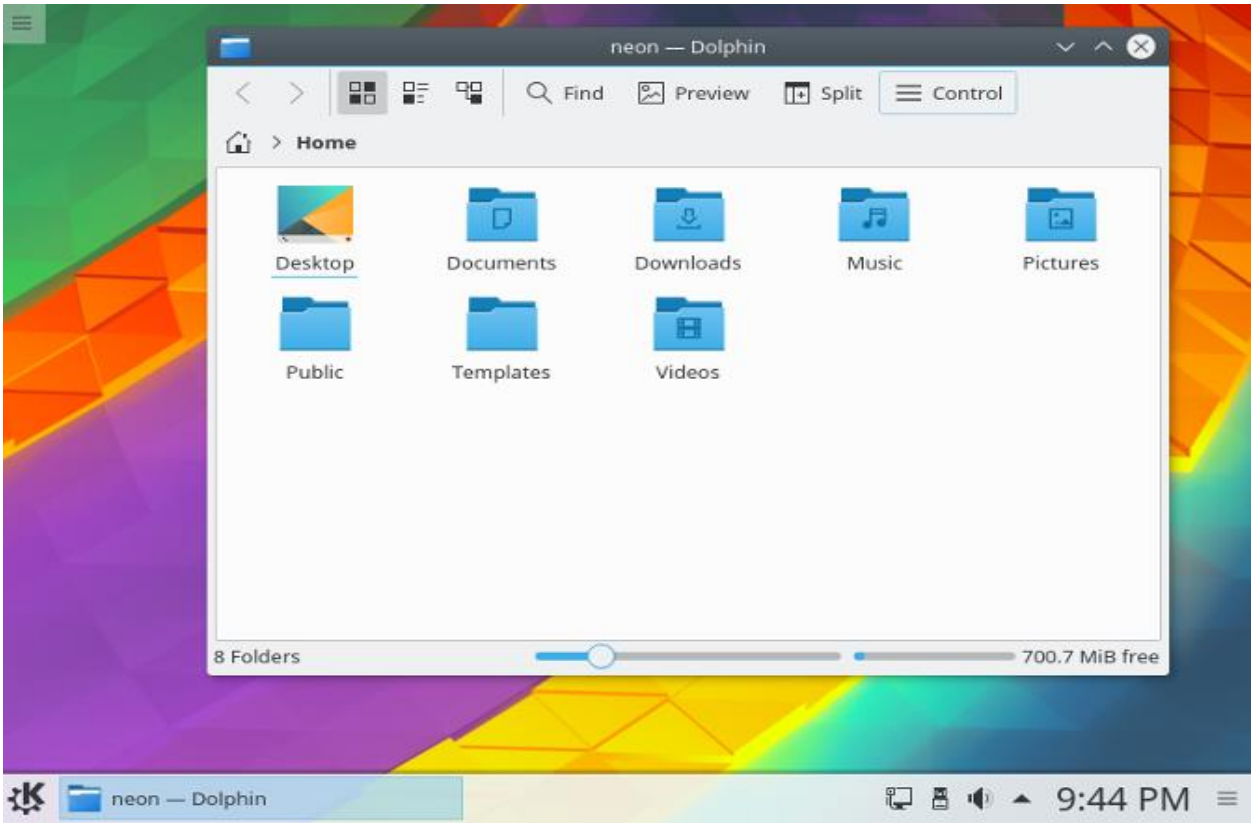
برای بیان بهتر این موضوع، چندین تصویر از محیط های دسکتاپی لینوکس یا desktop enviroment در ادامه آورده شده است. اینکه کدام بهتر است بسته به سلایق افراد تغییر می‌کند. شاید کسی از محیطی خوشش بیاید و همان محیط برای فرد دیگر مزخرف باشد.

GNOME 3

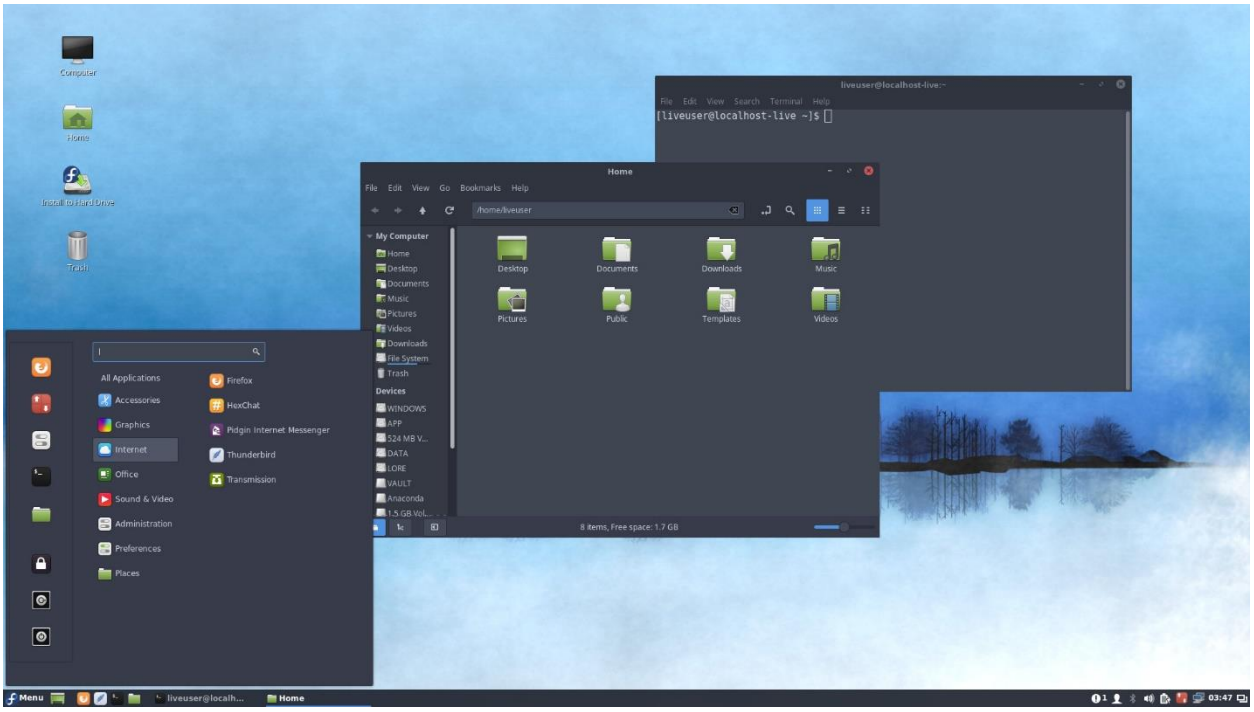


Ubuntu unity

KDE



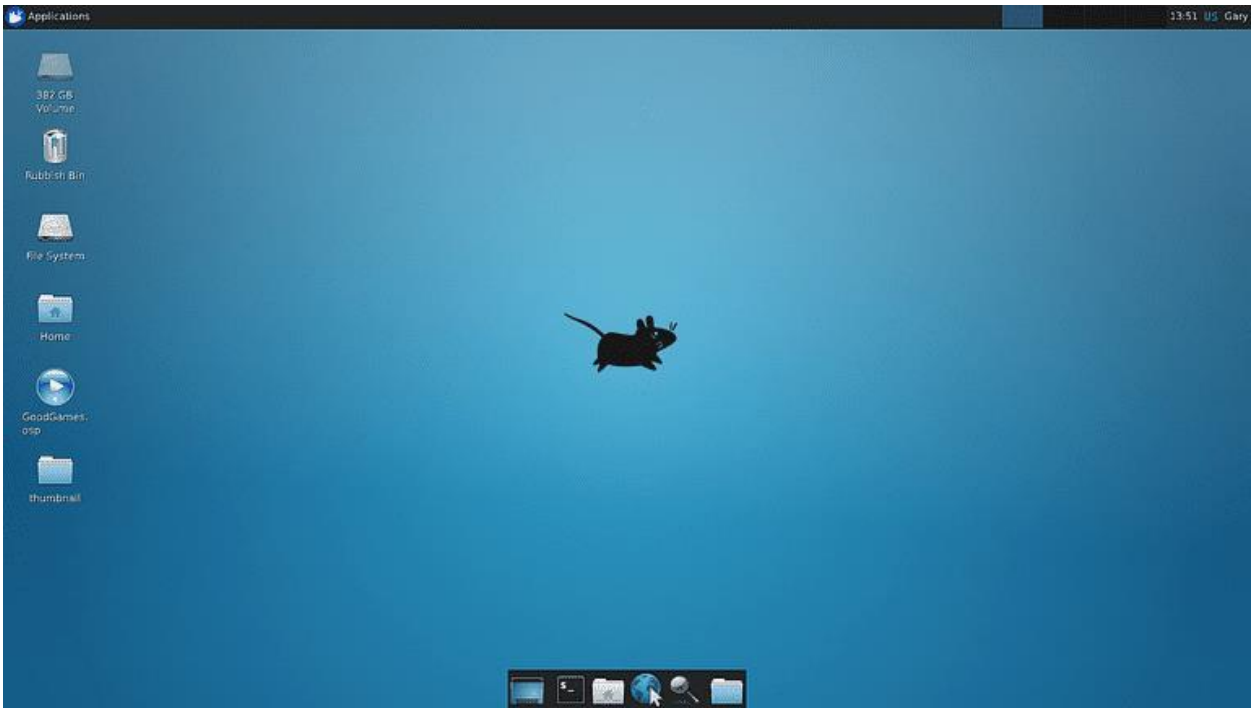
Cinnamon



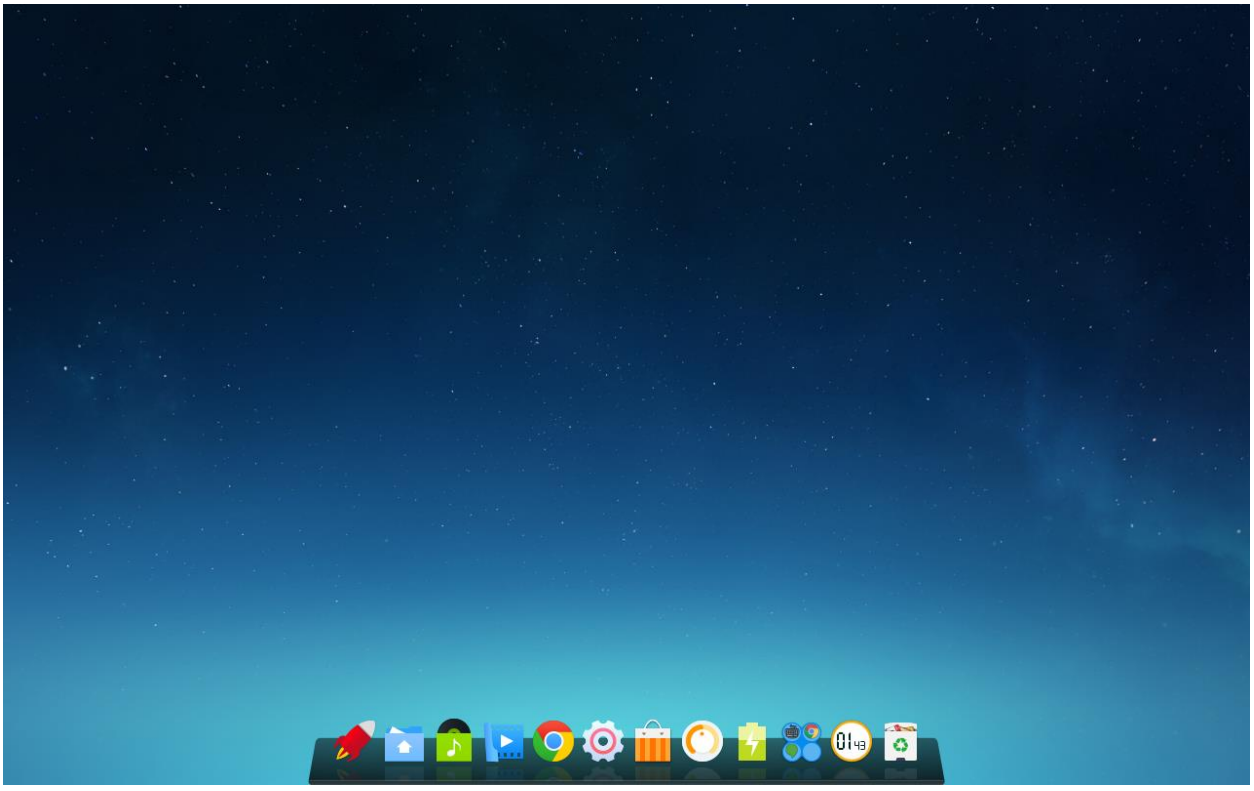
Enlightenment


Applications
Run Everything
Take Screenshot
Desktop
Windows
Settings
System
Settings Panel
Modules
Shelves
Theme
Wallpaper
All

Xfce

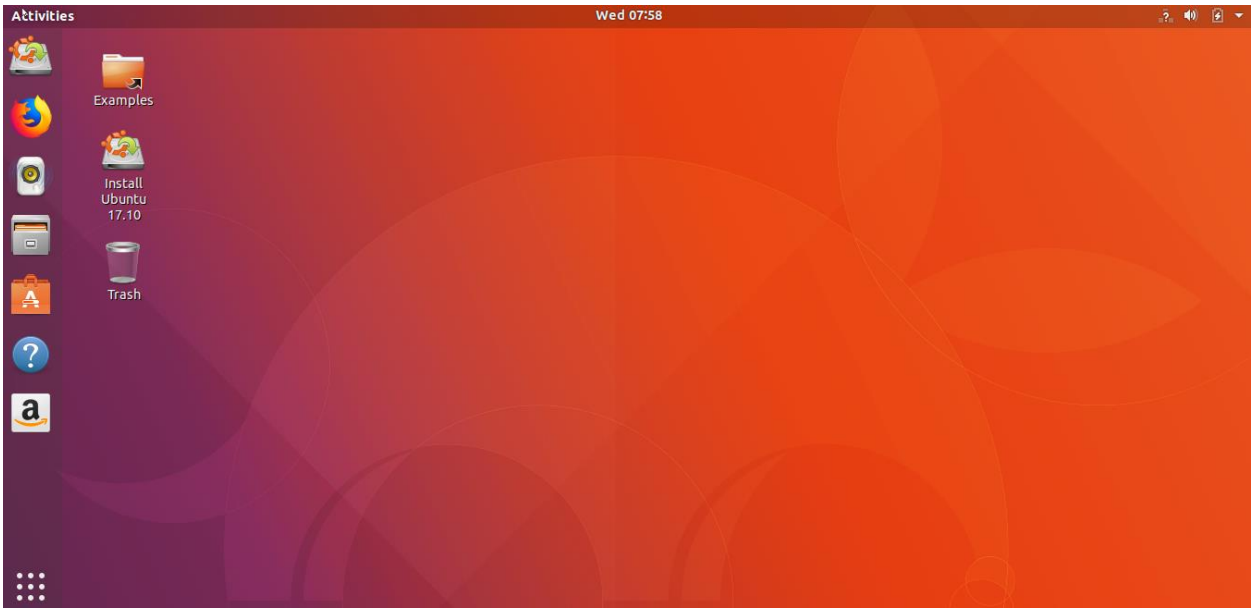

Applications
13:51 US Gary
382 GB Volume
Rubbish Bin
File System
Home
GoodGames.osp
thumbnail

Deepin



Pantheon

در نسخه اوبونتو 17، به جز اینکه ubuntu desktop با محیط Unity ارائه شده است، نسخه‌های مختلفی هم ارائه شده‌اند. مثل: Kubuntu که به جای محیط گرافیکی unity از KDE استفاده می‌کند و نسخه‌های دیگری مثل: ubuntu GNOME، Lubuntu، Xubuntu، ubuntu Budgie ، ubuntu studio،Ubuntu Kylin.



با کمی کارکردن با این محیط به راحتی می‌توانید شروع به استفاده از لینوکس کنید و کارهای ساده‌ای انجام دهید. استفاده از این محیط گرافیکی نیاز به توضیح خاصی ندارد. اما این محیط گرافیکی نمی‌تواند به تنهایی تمامی کارهای ما را انجام دهد و محدودیت‌هایی را دارد. پس باید از یک محیط قدرتمند دیگری استفاده کنیم.

کرنل (kernel)

می‌دانیم که در قلب سیستم‌عامل، کرنل حضور دارد. کرنل در واقع نرم‌افزاری است که به applicationها اجازه می‌دهد تا با سخت‌افزار یا Hardware کامپیوتر در ارتباط باشند و با سایر برنامه‌های درحال اجرا روی همان سیستم‌عامل همکاری کنند.

سخت‌افزار در واقع هر چیزی است که قابل لمس باشد مثل: موری و...

برای اینکه کرنل بتواند با سخت‌افزار کامپیوتر در ارتباط باشد باید چندین پیش‌نیاز را داشته باشد:

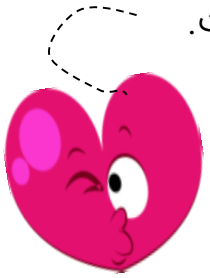
1- مطالبی را دربارهٔ درایورهای سخت‌افزاری واقع بر کامپیوتر را بداند: در لینوکس، کدی که به applicationها اجازه می‌دهد تا با هر قطعه سخت‌افزار در ارتباط باشند، یا در داخل کرنل بصورت درایور ساخته شده است یا بعد از اجرای کرنل با استفاده از چیزی به نام ماژوال قابل بارگذاری، به کرنل اضافه می‌شوند.
2- باید از عهده مدیریت applicationها بربیاد: کرنل باید تصمیم بگیرد که چه پردازش‌هایی الان در حال اجرا هستند و این پردازش‌ها به کدام پردازندهٔ کامپیوتر می‌توانند دسترسی داشته باشند و این دسترسی چقدر طول بکشد.
3- باید سیستم فایل‌ها و اجازهٔ دسترسی را تشخیص دهد: کرنل مسئولیت‌های سنگینی را به عهده دارد و چون مسئول این است که به پردازش‌ها اجازه خواندن (reading)، نوشتن (writing)، runشدن فایل‌هایی که در انواع مختلف سیستم‌فایل ذخیره شده‌اند را بدهد، پس باید ساختار این سیستم فایل‌ها را بداند. جدا از این باید اجازهٔ هر کاربر را بررسی کند تا بداند هر کاربر اجازه دسترسی به کدام فایل‌ها را دارد.

حواستو جمع کن از این به بعد خانوم آیندم اجازتو تعیین می کنه. دست از پا خطا کنی می‌دم شیفت دیلیتت کنه.

گیری کردیما.آرزو بر جوانان عیب نیست. چه حساب کردی روی خانوم آیندت!

حالا که توانایی لازم برای این مسئولیت بزرگ را به تنهایی نداریم (!) پس کرنل یک interface ای ارائه می‌دهد که با استفاده از آن توانایی این را داریم که با سخت‌افزار صحبت کنیم و ارتباط برقرار کنیم ولی غیرمستقیم.

کسی که این لطف را در حق ما می‌کند، shell نام دارد که رابط بین ما و کرنل است.



به قول بعضی‌ها، توی بلاد کفر پوشش می‌گن shell؛ فارسیش چی می‌شه؟ ترکی پیلیر سن؟

شِل یا پوسته هم پوشش می گن

Shell یک مفسر کامند لاینی (command line) است که به شما اجازه دسترسی به برخی از حیاتی‌ترین ابزارهای لینوکس را می‌دهد.

آخ چون کی شروع می‌شه؟ می‌خوام توشو بنویسم I love u و بفرستم برای عیال شاید ببینه

Shell روشی برای اجرای برنامه‌ها، کار با سیستم فایل، کامپایل کد کامپیوتر و مدیریت کامپیوتر ارائه می‌نماید. اگرچه shell از GUIهای مرسوم محبوبیت کمتری دارداما بخش زیادی از حرفه‌ای بودن یک فرد در لینوکس بستگی به استفاده او از کامندلاین‌ها در shell دارد. شاید در ابتدا قضیه کمی سخت جلوه کند اما اگر به درستی یاد بگیرید تبدیل به بزرگترین سرگرمی شما خواهد شد. شروع یک دنیای هیجان‌انگیز است که پایان ندارد.

Shellها انواع مختلفی دارند. گرافیکی، CLI که به آن‌ها text یا command lineهم گفته می‌شود. بسیاری از لینوکس‌ها هم shellگرافیکی دارند هم shell تکست.

قبلا هم اشاره شد که منظور از shellگرافیکی، همان دسکتاپی است که اغلب بعد لاگین کردن می‌بینید و گفتیم که این محیط، در هر سیستم لینوکسی‌ای می‌تواند متفاوت باشد. اما چیزی که می‌خواهیم اضافه کنیم این است که اصلا ممکن است سیستم لینوکسی شما محیط گرافیکی نداشته باشد مثل برخی از لینوکس سرورها. در این صورت باید حتما از text shell استفاده کنید. بنابراین تمرکز خود را روی یادگیری محیط‌های گرافیکی نگذارید. چون اصولا قدرت خاصی به شما نمی‌دهند.

چگونه به یک shell دسترسی داشته باشیم؟

این بستگی دارد به اینکه سیستم شما محیط گرافیکی یا اصطلاحا GUI دارد یا ندارد:

No desktop: (مثل برخی از لینوکس سرورها) بعد از لاگین‌کردن، به طور مستقیم کارتان با shell شروع می‌شود و نیازی به زحمت خاصی نیست.

With desktop: (مثل ubuntu که داریم استفاده می‌کنیم) باید terminal را باز کنید. (راست کلیک روی دسکتاپ و گزینه open terminal یا در بین applicationها پیدا کنید یا از شورت کات ctrl+alt+t استفاده کنید.)

Shell در واقع یک کلمهٔ کلی است؛ درست مثل زمانی که می‌گوییم مرورگر یا browser. یعنی یک مرورگر انواع مختلفی دارد مثل: کروم، فایرفاکس، اپرا و...

Shell نیز داستانی این چنینی دارد. یعنی shell یا پوسته، انواع مختلفی دارند اما shellای که به طور دیفالت روی همهٔ لینوکس‌ها حضور دارد، bash است. Bash مخفف Bourne Again Shell است. این نام از این حقیقت سرچشمه گرفته است که bash با اولین shell یونیکس سازگاری دارد. (یعنی پوستهٔ Bourneکه به وسیلهٔ دستور Sh نمایش داده می‌شد.)

حالا این حقیقت رو به ما نمی‌گفتی
می‌ترکیدی؟ گیجمون کردی

نرم‌افزار ترمینال (terminal)، نرم‌افزاری است که در محیط گرافیکی اجرا می‌شود و در پشت این پنجرهٔ ترمینال، bash در حال اجراست. پس هر چیزی را که در ترمینال تایپ کنید به bash می‌رساند. Bashهم آن را اجرا (run) کرده و نتیجه را در پنجرهٔ ترمینال به شما نشان می‌دهد.

خوبی ترمینال این است که چون در محیط‌های گرافیکی اجرا می‌شود توانایی این را داریم که اسکرول کنیم یا فونت و رنگ را عوض کنیم یا حتی چندین پنجرهٔ ترمینال باز کنیم.

پنجره ها رو باز کن عشق رو بیار به خونه تا که قناری عشق بخونه عاشقونه غمارو رها کن به فردا نگاه کن عشقو صدا کن... عشقو صدا کن

با خره گشتی؟

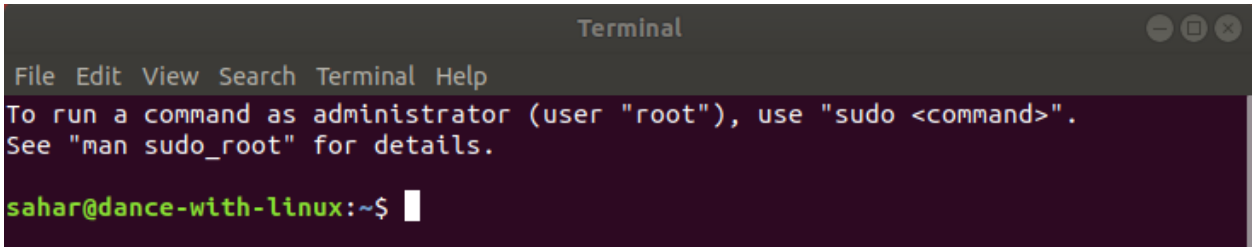
به طور دیفالت در اوبونتو، سه تا ترمینال نصب است:

1- Xterm
2- Uxterm
3- GNOME terminal

محیط‌های گرافیکی که base آن‌ها GNOME است (مثل اوبونتو) ترمینال دیفالت آن‌ها GNOME terminal است.

لینوکس‌هایی که محیط گرافیکی آن‌ها KDE است، ترمینال دیفالت آن‌ها، کنسول نام دارد و هر کدام ویژگی‌های خاص خود را دارند.

بعد از اینکه shell را بالا آوردید (مثلا ترمینال در اوبونتو)، اولین چیزی که می‌بینید، خط اعلان یا prompt نام دارد.



علامت prompt برای یک کاربر عادی، فقط یک علامت دلار یا $ است. قبلا هم گفته بودیم که کاربر عادی قابلیت مدیریت سیستم را ندارد و تنها می‌تواند از سیستم استفاده کند.

علامت prompt برای یک کاربر root یا super user، علامت # است و می‌تواند به فایل‌های حیاتی دسترسی داشته باشد(کمی خطرناک است(!))

این کاراکترها و علامت‌ها را خودتان هم می‌توانید تغییر دهید.

من می‌خوام عکس سه در چهار لینوکس رو بذارم.
پول که نداریم توی کیف پولمون بذاریم حداقل اینجا بذاریم

Bash چگونه کار می‌کند؟

شما به عنوان لینوکس کار که بیشتر وقت خود را پشت این سیستم‌عامل می‌گذرانید بهتر است تا حدودی به خوبی درک کنید که پشت پردهٔ چه اتفاقاتی می‌افتد.

وقتی یک command line یا اصطلاحا دستوری را در ترمینال (terminal) تایپ می‌کنید، بعد از زدن enter یا return، bash آن‌را اجرا (run) می‌کند و خروجی را به شما نشان می‌دهد.

Bash تنها نخستین کلمه در command line را به عنوان کامند یا دستور در نظر می‌گیرد. منظور از نخستین کلمه، یعنی تا وقتی که شما کلید space را فشار دهید. باقی command یا دستور را به عنوان آرگمان در نظر می‌گیرد.

وقتی یک دستور را وارد کردید و bash اجرا کرد، در پایان کار خود دوباره این خط prompt را به شما نشان می‌دهد. اگر نشان نداد یعنی هنوز مشغول اجرای کامند قبلی است.

برای شروع و آشنایی با shell برخی از این کامندها را بررسی می‌کنیم.

بررسی لاگین خود

وقتی به یک سیستم لینوکس login می‌کنید، لینوکس شما را به عنوان یک موجودیت خاص نمایش می‌دهد یا به عبارتی دیگر به شما یک identity تعلق می‌گیرد. این identity شامل نام کاربری(user id)، نام گروه شما (group id) می‌باشد. برای نمایش identity خودتان دستور id را تایپ کنید.

```
sahar@dance-with-linux:~$ id
uid=1000(sahar) gid=1000(sahar) groups=1000(sahar),4(adm),24(cdrom),27(sudo),30(
dip),46(plugdev),118(lpadmin),128(sambashare)
```

این خطوط نشان می‌دهد که نام کاربر sahar بوده که به وسیلهٔ عددی کاربر 1000(uid) نشان داده است. در اینجا گروه اصلی که sahar به آن تعلق دارد در groups نوشته شده. همچنین sahar به گروه‌های دیگری مثل adm(gid 4) و بقیه گروه‌ها که آمده، تعلق دارد.

این اسامی و اعداد نشان‌دهندهٔ permissionها یا اجازه‌هایی است که sahar برای دسترسی به منابع کامپیوتر دارد.

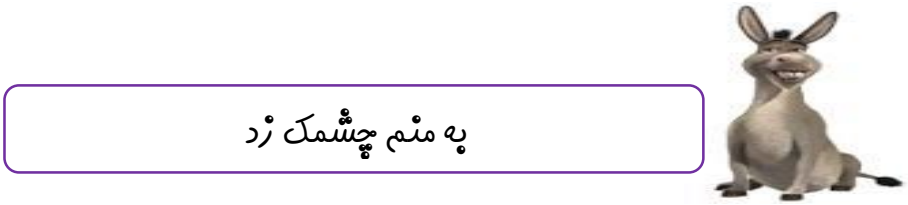
فهمیدن shell دیفالت

قبلا اشاره کرده بودیم که shell دیفالت لینوکس‌ها با هم تفاوت دارد. برای اینکه بدانید shell دیفالت یوزر شما چیست از این دستور استفاده کنید:

```
sahar@dance-with-linux:~$ who
sahar    tty2         2018-05-09 19:28 (/dev/tty2)
sahar@dance-with-linux:~$ grep sahar /etc/passwd
sahar:x:1000:1000:sahar,,,:/home/sahar:/bin/bash
sahar@dance-with-linux:~$
```

دستور who لاگین فعلی شما را نمایش می‌دهد و دستور grep تعریف اکانت شما را در فایل /etc/passwd نشان می‌دهد. در اینجا بجای sahar باید user خودتان را جایگزین کنید و در ادامه /etc/passwd را تایپ کنید. در فیلد آخر، /bin/bash، نشان می‌دهد که shell دیفالت شما bash می‌باشد.

(همان طور که می‌بینید، بعد از اجرای کامند، دوباره خط اعلان یا prompt به شکل چشمک زنان نشان داده شده است. یعنی کامند قبلی اجرا شده و حالا منتظر گرفتن دستور جدید است.)



با این حال، این امکان وجود دارد که با shellهای دیگری نیز کار کنید(با فرض اینکه نصب شده‌اند یا اگر نصب نشده‌اند می‌توانید نصب کنید.) Shellهای زیادی وجود دارد مثل ksh، tcsh، sh،csh ، dash و...

برای کار با این shellها کافی است اسم آن‌ها را وارد کنید و وقتی هم که کارتان با آن shell تمام شد، می‌توانید با تایپ دستور exit برگردید به shell دیفالت. (در اینجا bash دیفالت است.)

وقتی بفهمید یک shell چگونه کار می‌کند، به راحتی کار با shellهای دیگر را یاد می‌گیرید. یا می‌توانید از صفحهٔ راهنمای آن shell استفاده کنید برای مثال با تایپ man bash لیستی از کامندها و مشخصات آن shell مورد نظر را می‌توانید پیدا کنید.

بررسی دایرکتوری‌ها

برای هر shell مکانی در سیستم فایل لینوکس وجود دارد که به current directory یا working directory یا دایرکتوری کنونی شناخته می‌شود مثلا وقتی برای اولین بار به داخل لینوکس لاگین می‌کنید، کار با دایرکتوری home را آغاز می‌کنید.

این دایرکتوری به چه دردی می‌خوره؟

وقتی درخواست بازکردن یا ذخیره‌سازی یک فایل را می‌کنید، shellشما از دایرکتوری کنونی یا همان current directory به عنوان نقطهٔ رجوع استفاده می‌کند. مثلا برای saveکردن یک فایل، کافی است نام فایل را بدهید و آن فایل در دایرکتوری کنونی شما ذخیره خواهد شد.

برای اینکه بفهمید دایرکتوری کنونی شما چیست، pwd را تایپ کنید:

```
sahar@dance-with-linux:~$ pwd
/home/sahar
```

دستور pwd دایرکتوری کنونی شما را نشان می‌دهد. دایرکتوری فعلی در اینجا /home/sahar می‌باشد. هم‌چنین اگر دایرکتوری کنونی را تغییر دادید می‌توانید با دستور cd که مخفف change directory است به دایرکتوری home باز گردید. اگر به دنبال این دستور، نام یک دایرکتوری آورده شود، دایرکتوری کنونی شما به آن دایرکتوری انتخابی تغییر می‌کند ولی دستوری cd بدون هیچ دنباله‌ای شما را به دایرکتوری home می‌برد.

دستور ls لیست محتویات موجود در دایرکتوری فعلی شما را نشان می‌دهد.

```
sahar@dance-with-linux:~$ ls
Desktop    Downloads         Music     Public     Videos
Documents  examples.desktop  Pictures  Templates
sahar@dance-with-linux:~$
```

چجوری از یک shell خارج شویم؟

بعد از اینکه کارتان با shell به پایان رسید، با تایپ دستور exit یا ctrl+d می‌توانید خارج شوید. این‌ها برای دست گرمی بود وگرنه صدها دستور دیگر در دایرکتوری‌هایی مثل /bin یا /usr/bin وجود دارد. همچنین دستوراتی برای مدیریت سیستم در داخل دایرکتوری‌های /sbin و یا /usr/sbin قرار دارند. در ادامه کتاب بیشتر در مورد دستورات صحبت می‌کنیم.

نحوه نوشتن کامندها یا command syntax

برخی از commandها، هیچ آرگمان و یا آپشنی در دنبالهٔ خودشان ندارند. مثل: ls یا pwd

می‌توانیم یکسری آیتم‌ها به این دستورها اضافه کنیم. با این کار چگونگی کارکرد آن دستور تغییر می‌کند. این آیتم‌ها عبارت است از:

یه حسی بهم می‌گه با این آیتم‌ها، کامندهامون شناسی بلند می‌شه یا یه چیزی مثل اسفناجی که ملوان زپل می‌خورد البته امروزه علم اثبات کرده اون اسفناج نبوده که اون می‌خورده. هر چیه همون چیزی بوده که این خره هم می‌خوره

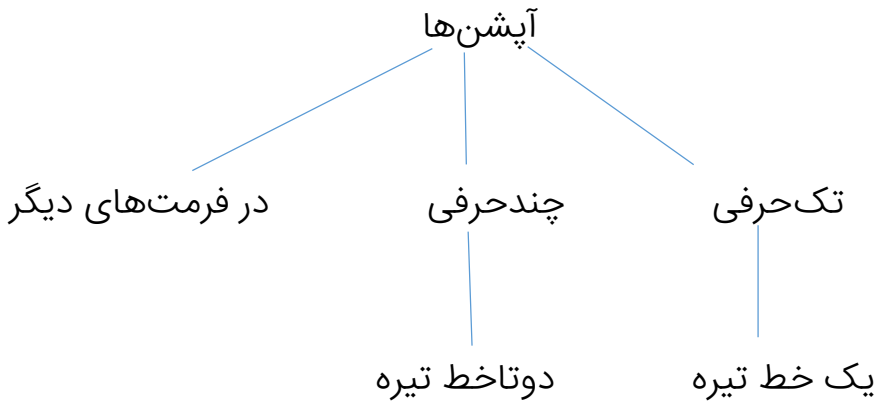
1- آپشن‌ها (options)
2- آرگمان‌ها (arguments)
3- متغیرهای محیطی (enviroment variables)
4- متاکاراکترها (metacharacters)

آپشن‌ها (options)

با آپشن‌ها می‌توانیم رفتار یک کامند را تغییر دهیم. به کامند دقیقا مشخص کنیم که چه می‌خواهیم، چه نمی‌خواهیم، چه کاری بکند، چه کاری نکند اکثر کامندها دارای یک یا چندین آپشن هستند.

آپشن‌های یک کامند را برنامه‌نویسی که آن کامند را نوشته، مشخص می‌کند.

همه کامندها آپشن دارند منتهی در فرمت‌های مختلف. مثلا برخی از برنامه‌ها، آپشنش یک خط تیره یا dash یا – دارد، برخی دیگر دوتا خط تیره دارد و یا اصلا برخی ندارند.



ولی در کل، آپشن‌ها یک لباسی دارند (یک الگویی دارند) که اکثر کامندها آن‌ها را می‌پوشند. در ادامه به شرح این لباس می‌پردازیم.

آپشن‌های تک‌حرفی معمولا با یک خط تیره یا dash شروع می‌شوند.

برای مثال پیش از این اشاره کردیم که دستور ls محتویات دایرکتوری فعلی را نشان می‌دهد،اگر به دنبال این دستور آپشن l– (خط تیره اِل) تایپ کنیم، لیست کاملی از اطلاعات

با جزئیات را ارائه می‌کند و همچنین تایپ آپشن a– فایل‌های مخفی (فایل‌های نقطه) را نشان می‌دهد.

با این کتاب نوشتنت. فکر کنم تا حالا هیشکی ازت جزوه نگرفته. آخه خط تیره ال؟؟؟؟ تمام پاورهامو ریختی بهم

```
sahar@dance-with-linux:~$ ls -l -a
total 1528
drwxr-xr-x 16 sahar sahar    4096 Jun  5 07:40 .
drwxr-xr-x  3 root  root     4096 May  9 12:14 ..
-rw-r--r--  1 sahar sahar 1454080 May 11 04:46 backup.tar
-rw-------  1 sahar sahar     748 May 29 02:09 .bash_history
-rw-r--r--  1 sahar sahar     220 May  9 12:14 .bash_logout
-rw-r--r--  1 sahar sahar    3771 May  9 12:14 .bashrc
drwx------ 14 sahar sahar    4096 May 29 00:03 .cache
drwxr-xr-x 16 sahar sahar    4096 May 29 01:05 .config
drwxr-xr-x  2 sahar sahar    4096 May 11 05:45 Desktop
drwxr-xr-x  2 sahar sahar    4096 May  9 12:42 Documents
drwxr-xr-x  2 sahar sahar    4096 May  9 12:42 Downloads
-rw-r--r--  1 sahar sahar    8980 May  9 12:14 examples.desktop
drwx------  2 sahar sahar    4096 May 29 00:03 .gnupg
-rw-------  1 sahar sahar    3496 Jun  5 07:39 .ICEauthority
drwxr-xr-x  3 sahar sahar    4096 May  9 12:42 .local
drwx------  5 sahar sahar    4096 May 29 00:01 .mozilla
```

توضیح درباره شکل:

دایرکتوری‌هایی که یک نقطه (.) دارند نشان‌دهندهٔ دایرکتوری /home/sahar هستند یا همان /home/user یعنی دایرکتوری فعلی است. و آن‌هایی که دو نقطه (..) دارند، اصطلاحا دایرکتوری والد گفته می‌شود. این دایرکتوری والد، همان /home است که تنها در اختیار کاربر root می‌باشد.تمامی فایل‌های دیگر در اختیار sahar نیز هست.

(اسامی فایل‌ها یا دایرکتوری‌هایی که در سمت راست آن‌ها یک نقطه (.) قرار دارد، فایل‌هایی را نشان می‌دهند که برای ذخیره‌سازی خاصیت‌های GUI یا خاصیت‌های shell به کار برده می‌شوند.)

در کل، فایل‌هایی که hidden (مخفی) هستند با یک نقطه شروع می‌شوند. آوردن آپشن a– می‌گوید فایل‌های hidden را هم نشان بده.

بقیه اطلاعات لیست مربوط به اندازه هر فایل بر حسب بایت، تاریخ و زمان آخرین باری که دستتون به فایل خورده رو نشون می‌ده. باتریش نکشید اینارم بگه

l– (دَش‌ال) در کامند ls معنای long list را می‌دهد.

ممکن است در برنامه‌ای دیگر همین l– معنی دیگری، مثلا login بدهد و کلا کار دیگری انجام دهد. یعنی آپشن‌ها بسته به چیزهایی که برنامه‌نویس نوشته است، می‌توانند متفاوت باشند و کاربردهای مختلفی داشته باشند.

اونجا که گفتی "برنامه‌ای دیگر" فکر کردم می‌خوای مارو به خدای بزرگ بسپاری

خب می‌خوای من بسپارمت؟

تا برنامه‌ای دیگر شما رو به خدای بزرگ می‌سپارم

می‌توانید چندین آپشن تک حرفی را در کنار یکدیگر بعد از یک خط تیره تایپ کنید. برای مثال : la-

می‌خواد بگه می‌شه آپشن‌های تک حرفی رو توی یه مخلوط کن بریزید و قاطیش کنید و کنار هم بیارید مثل ls –la

یعنی این سه تا هیچ فرقی با هم ندارند:

ls –l –a

ls –a –l

ls –la

```
sahar@dance-with-linux:~$ ls -la
total 84
drwxr-xr-x 13 sahar sahar 4096 May  9 12:45 .
drwxr-xr-x  3 root  root  4096 May  9 12:14 ..
-rw-------  1 sahar sahar   14 May  9 12:45 .bash_history
-rw-r--r--  1 sahar sahar  220 May  9 12:14 .bash_logout
-rw-r--r--  1 sahar sahar 3771 May  9 12:14 .bashrc
drwx------  9 sahar sahar 4096 May  9 12:43 .cache
drwxr-xr-x 11 sahar sahar 4096 May  9 12:43 .config
drwxr-xr-x  2 sahar sahar 4096 May  9 12:42 Desktop
drwxr-xr-x  2 sahar sahar 4096 May  9 12:42 Documents
drwxr-xr-x  2 sahar sahar 4096 May  9 12:42 Downloads
-rw-r--r--  1 sahar sahar 8980 May  9 12:14 examples.desktop
-rw-------  1 sahar sahar  310 May  9 12:42 .ICEauthority
drwxr-xr-x  3 sahar sahar 4096 May  9 12:42 .local
drwxr-xr-x  2 sahar sahar 4096 May  9 12:42 Music
drwxr-xr-x  2 sahar sahar 4096 May  9 12:42 Pictures
-rw-r--r--  1 sahar sahar  675 May  9 12:14 .profile
drwxr-xr-x  2 sahar sahar 4096 May  9 12:42 Public
drwxr-xr-x  2 sahar sahar 4096 May  9 12:42 Templates
drwxr-xr-x  2 sahar sahar 4096 May  9 12:42 Videos
```

اما تعداد آپشن‌های تک حرفی کم است، پس اگر کامندی نیاز به آپشن بیشتر داشت چه کنیم؟

در آن صورت، برنامه‌نویس یکسری آپشن‌هایی در نظر می‌گیرد که به آن آپشن‌های long گفته می‌شود. این آپشن‌های لانگ، دوتا خط تیره دارند. چرا؟ چون اگر یک خط تیره باشد، هر حرف آن آپشن را، یک آپشن مجزا در نظر می‌گیرد. ما با دوتا خط تیره گذاشتن به او می‌فهمانیم که چیزی که نوشتیم همگی باهم معرف یک آپشن است. مثل ls --help

اگر دوتا خط تیره تایپ نشود، help به صورت جداگانه h، e، l، p، حساب می‌شود.

خدا لگدش کنه که همه رو تیکه تیکه می‌کنه

مثلا فرض کنید نمی‌دانیم دستور ls چه کاری انجام می‌دهد. می‌توانیم از آپشن help کمک بگیریم:

```
sahar@dance-with-linux:~$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not ignore entries starting with .
  -A, --almost-all           do not list implied . and ..
      --author               with -l, print the author of each file
  -b, --escape               print C-style escapes for nongraphic characters
      --block-size=SIZE      scale sizes by SIZE before printing them; e.g.,
                               '--block-size=M' prints sizes in units of
```

در اینجا آپشن‌هایی که کامند ls می‌تواند بپذیرد را نشان داده است.

با تایپ less خروجی به طور صفحه به صفحه نشان داده خواهد شد. برای خروج می‌توانید q را فشار دهید:

```
sahar@dance-with-linux:~$ ls --help | less
```

```
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not ignore entries starting with .
  -A, --almost-all           do not list implied . and ..
      --author               with -l, print the author of each file
  -b, --escape               print C-style escapes for nongraphic characters
      --block-size=SIZE      scale sizes by SIZE before printing them; e.g.,
                               '--block-size=M' prints sizes in units of
                               1,048,576 bytes; see SIZE format below
  -B, --ignore-backups       do not list implied entries ending with ~
  -c                         with -lt: sort by, and show, ctime (time of last
                               modification of file status information);
                               with -l: show ctime and sort by name;
                               otherwise: sort by ctime, newest first
  -C                         list entries by columns
      --color[=WHEN]         colorize the output; WHEN can be 'always' (default
                               if omitted), 'auto', or 'never'; more info below
  -d, --directory            list directories themselves, not their contents
  -D, --dired                generate output designed for Emacs' dired mode
  -f                         do not sort, enable -aU, disable -ls --color
:
```

آپشن‌هایی که مقدار می‌پذیرند

برخی از آپشن‌ها نیز وجود دارند که به وسیله آن‌ها ما "مقداری" یا یک "value" ای به پراسس پاس می‌دهیم.

مثلا کامند head با آرگمان /etc/samba/smb.conf با آپشن n– چند خط اول فایل smb.conf را نشان می‌دهد. برای این که بگوییم چند خط را نشان دهد، عدد یا آن value را در کنار n می‌نویسیم.

مثلا من یک فایل تکستی در دایرکتوری /HOME/SAHAR ساختم به اسم EXAMPLE.TXT و داخل آن متنی گذاشتم. می‌خواهم با استفاده از کامند HEAD تنها دو خط اول این فایل را بخوانم:

```
sahar@dance-with-linux:~$ head /home/sahar/example.txt -n2
Dance with linux
learn gnu/linux
sahar@dance-with-linux:~$
```

اگر بخواهم شش خط اول را بخوانم:

```
sahar@dance-with-linux:~$ head /home/sahar/example.txt -n6
Dance with linux
learn gnu/linux
the fun way
sahar shaker
web:
saharshaker.com
sahar@dance-with-linux:~$
```

یعنی با آوردن آپشن n– و بعد مقدار عددی 6 مشخص کردم که شش خط اول را نشان بده.

در برخی آپشن‌ها این value به آپشن (اینجا n) می‌چسبد ولی در برخی هم یک اسپیس بین آن‌ها وجود دارد. در برخی مواقع هم باید یک مساوی قرار بگیرد: n=6-

ولی این وسط داریم کامندهایی را که سرکشی کردند یا به عبارتی حال کرده‌اند طوری دیگری آپشن داشته باشند. مثلا کامند ip آپشنی ندارد که با dash بتوان با او رفتار کرد. یعنی آپشن‌های ip هیچکدام خط تیره یا dash ندارد.

اگر بخواهیم ip یک interfaceای را ببینیم، ip addr show dev lo

```
sahar@dance-with-linux:~$ ip addr show dev lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
sahar@dance-with-linux:~$
```

توی اینجا بعد از ip هر چه آمده، آپشن است اما هیچکدام خط تیره ندارد!

همه این‌ها رو باید حفظ کنیم؟ 0_o

نه. بلکه بیشتر باید به help آن کامند بروید تا متوجه شوید که چه آپشن‌هایی و به چه شکلی وجود دارد؛ رفته رفته با تکرار یادتان می‌ماند، مثل اولین باری که مشتق می‌گرفتید فرمول حفظ می‌کردید ولی بعدها ذهنی انجام می‌دهید.

مچتو گرفتم. مشتق کیه؟

منتهی اکثر کامندها، آپشن‌هایشان همین لباس یدونه dash یا دوتا dash را پوشیده‌اند.

پیداکردن اطلاعات در مورد کامندها

دستوراتی دیگری که اطلاعات یا information در مورد کامندها در اختیار شما می‌گذارند، یکی man و دیگری info است. اما کامند help خلاصه‌ای از اطلاعات دستورات را نشان می‌دهد. پیش از این از آپشن helpهم استفاده کردیم.

وقتی اولین بار از shell استفاده می‌کنید، شاید فکر کنید هیولاست! شما فقط یک خط prompt می‌بینید که منتظر است تا دستورات شما را اجرا کند.

خیلی هم دلش بخواد هیولا باشه

از کجا بفهمیم اصلا چه کامندهایی هست؟ چه آپشن‌هایی می‌گیرن؟ یا چجوری پیشرفته‌تر ازشون استفاده کنیم و افزودنی‌های مجازشون چیه؟

مگه کلاس آشپزی یا مانع است؟ بقیه قصه برای جواب سوالته

1- چک‌کردن PATH

در shell تایپ کنید:

```
echo $PATH
```

نشد

بزرگ تایپ کن PATH قبلشم یه علامت دلار بذار.

Every body listen and repeat PATH

یک لیست از دایرکتوری‌های یا اتاق‌هایی که کامندها در آن حضور دارند به شما نشان داده‌خواهد شد. محتویات داخل این دایرکتوری‌ها اکثرا دستورات استاندارد لینوکس را

نشان می‌دهد، یعنی داخل این اتاق‌ها، دستورات حضور دارند. مثلا می‌خواهیم بفهمیم در دایرکتوری یا اتاق bin چه دستوراتی زندگی می‌کنند:

```
sahar@dance-with-linux:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loc
al/games:/snap/bin
sahar@dance-with-linux:~$ ls /bin
bash            fusermount      ntfs-3g         su
bunzip2         getfacl         ntfs-3g.probe   sync
busybox         grep            ntfscat         systemctl
bzcat           gunzip          ntfscluster     systemd
bzcmp           gzexe           ntfscmp         systemd-ask-password
bzdiff          gzip            ntfsfallocate   systemd-escape
bzegrep         hciconfig       ntfsfix         systemd-hwdb
bzexe           hostname        ntfsinfo        systemd-inhibit
bzfgrep         ip              ntfsls          systemd-machine-id-setup
bzgrep          journalctl      ntfsmove        systemd-notify
bzip2           kbd_mode        ntfsrecover     systemd-sysusers
bzip2recover    kill            ntfssecaudit    systemd-tmpfiles
bzless          kmod            ntfstruncate    systemd-tty-ask-password-agent
bzmore          less            ntfsusermap     tar
cat             lessecho        ntfswipe        tempfile
chacl           lessfile        open            touch
chgrp           lesskey         openvt          true
chmod           lesspipe        pidof           udevadm
chown           ln              ping            ulockmgr_server
chvt            loadkeys        ping4           umount
```

2- استفاده از کامند help

بعضی از کامندها هم در دایرکتوری‌ها یا همان اتاق‌ها دیده نمی‌شوند. در واقع این کامندها داخل خود shell از پیش ساخته شدند. برای دیدنش از کامند help می‌توانید استفاده کنید. منتهی خروجی این کامند چون خیلی زیاده، می‌توانید با نوشتن کامند زیر، به صورت صفحه به صفحه ببینید.

help | less

```
Use `info bash' to find out more about the shell in general.
Use `man -k' or `info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

 job_spec [&]                            history [-c] [-d offset] [n] or hist>
 (( expression ))                        if COMMANDS; then COMMANDS; [ elif C>
 . filename [arguments]                  jobs [-lnprs] [jobspec ...] or jobs >
 :                                       kill [-s sigspec | -n signum | -sigs>
 [ arg... ]                              let arg [arg ...]
 [[ expression ]]                        local [option] name[=value] ...
 alias [-p] [name[=value] ... ]          logout [n]
 bg [job_spec ...]                       mapfile [-d delim] [-n count] [-O or>
 bind [-lpsvPSVX] [-m keymap] [-f file>  popd [-n] [+N | -N]
 break [n]                               printf [-v var] format [arguments]
 builtin [shell-builtin [arg ...]]       pushd [-n] [+N | -N | dir]
 caller [expr]                           pwd [-LP]
 case WORD in [PATTERN [| PATTERN]...)>  read [-ers] [-a array] [-d delim] [->
 cd [-L|[-P [-e]] [-@]] [dir]            readarray [-n count] [-O origin] [-s>
 command [-pVv] command [arg ...]        readonly [-aAf] [name[=value] ...] o>
 compgen [-abcdefgjksuv] [-o option] [>  return [n]
 complete [-abcdefgjksuv] [-pr] [-DE] >  select NAME [in WORDS ... ;] do COMM>
 compopt [-o|+o option] [-DE] [name ..>  set [-abefhkmnptuvxBCHP] [-o option->
```

قبول نیست چیزایی که ما نمی‌فهمیم رو بنویسی و ما فقط تایپ کنیم. اصلا من بازی نمی‌کنم

قرار نیست اینجوری تموم شه، وگرنه غم‌انگیزه. همشو می‌گه اگه خدا قبول کنه. تو فقط تحمل داشته باش. اینا رو یاد بگیر که جایی موندی هی گیج نشی

یعنی چی هی می‌گه صفحه‌صفحه؟

یعنی موستو اسکرول کنی میاد پایین بالا، می‌تونی q رو بزنی تموم کنی این بازی کثیفو

این دستور فقط با shell یا پوستهٔ bash کار می‌کند اگر با shell دیگری کار می‌کنید، این دستور اجرا نخواهد شد.

3- می‌توانید help– را با دستوری که می‌خواهید به کار ببرید

خیلی از کامندهایی که وجود دارند، آپشنی به اسم help هم گذاشته‌اند تا کاربر گیج نشود. می‌توانید کامندی را که می‌خواهید بغلش help-- را استفاده کنید.

مثلا فرض کنید نمی‌دانیم ls چه آپشن‌هایی را قبول می‌کند:

```
sahar@dance-with-linux:~$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not ignore entries starting with .
  -A, --almost-all           do not list implied . and ..
      --author               with -l, print the author of each file
  -b, --escape               print C-style escapes for nongraphic characters
      --block-size=SIZE      scale sizes by SIZE before printing them; e.g.,
                               '--block-size=M' prints sizes in units of
                               1,048,576 bytes; see SIZE format below
  -B, --ignore-backups       do not list implied entries ending with ~
  -c                         with -lt: sort by, and show, ctime (time of last
                               modification of file status information);
                               with -l: show ctime and sort by name;
                               otherwise: sort by ctime, newest first
  -C                         list entries by columns
      --color[=WHEN]         colorize the output; WHEN can be 'always' (default
                               if omitted), 'auto', or 'never'; more info below
  -d, --directory            list directories themselves, not their contents
  -D, --dired                generate output designed for Emacs' dired mode
  -f                         do not sort, enable -aU, disable -ls --color
```

من این ls رو توی خوابمم می‌بینم. جز ls کامند دیگه‌ای بلد نیستی؟ یکم خوشحالمون کن دیگه چیه همش تکرار؟ تازه بالا هم اینو مثال زدی.

تو دیگه از تنوع حرف نزن. توی این هزار سالی که از خدا عمر گرفتم فقط قیافت همین‌شکلی بود. ما هر شوخی‌ای کردیم دریغ از یه انحنا روی صورتت

هزاااااااااار سالته؟؟؟؟ خجالت بکش از هیکلت.

بابام خدابیامرز عادت داشت ما هر خریتی که می کردیم می گفت چهل سالته از هیکلت خجالت بکش. اگر تو رو می‌دید که هزاره‌ای، خیلی خر کیف می‌شد.

باباتم مُرد و تو رو برای ما به ارث گذاشت، همهٔ خریت‌ها نسل به نسل چرخیده و انباشته شده توی تو. تو هم خراب شدی روی سر ما. یه‌ذره از خریتتو با خودش به گور نبرد یا ارث می‌ذاشت برای بقیه.

مثلا با دستور mkdir می‌توان یک دایرکتوری ساخت، برای اینکه اطلاعات بیشتری از او داشته باشیم:

این دستور آمار همه رو می‌ده؟ بزن ببینیم خونه لینوکس کجاست؟

```
sahar@dance-with-linux:~$ mkdir --help
Usage: mkdir [OPTION]... DIRECTORY...
Create the DIRECTORY(ies), if they do not already exist.

Mandatory arguments to long options are mandatory for short options too.
  -m, --mode=MODE   set file mode (as in chmod), not a=rwx - umask
  -p, --parents     no error if existing, make parent directories as needed
  -v, --verbose     print a message for each created directory
  -Z                   set SELinux security context of each created directory
                         to the default type
      --context[=CTX]  like -Z, or if CTX is specified then set the SELinux
                         or SMACK security context to CTX
      --help     display this help and exit
      --version  output version information and exit

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/mkdir>
or available locally via: info '(coreutils) mkdir invocation'
sahar@dance-with-linux:~$
```

توی واقعیت برای خونه ساختن پاس عین این خر پول جمع کنی، صد نفرو راضی کنی بیان بسازن بعد اینجا با یه mkdir یه اتاق می‌زنی زندگیتو می کنی

مگه شما خرها هم خونه می‌سازین؟

آره مگه مثل شما آواره سحر نامیم؟ ما خونه زندگی می‌سازیم روی پای خودمون وامیسیم. بیرون از اینجا واقعیت محضه

کم مونده بود تو یکی یهمون درس اخلاق بدی

4- دستور man

دستورات قبلی، خلاصه‌ای از اطلاعات را در اختیار شما می‌گذارند. برای اطلاعات بیشتر از دستور man استفاده کنید.

نکنین. خانواده نشسته. نشون می‌ده لباس چی پوشیده.

می‌خواست بگه برای اطلاعات بیشتر به روزنامه‌های کثیرالانتشار مراجعه کنید. این همش می‌خواد مارو بپیچونه. حالا ببین کی گفتم

یعنی چی هی کلمه man کنار کامند مردم بیاد. ملت ناموس دارن. چه وضعشه.

تایپ کردم این man رو.

می‌گه what manal page do you want?

وقتشه برم بیلمو بیارم؛

باید کامندی که می‌خوای رو هم بغلش بیاری. man command

برای مثال بزنید man mkdir

دوباره قفل کرد روی یه دستور جدید. پاپا تنووووووع

```
MKDIR(1)                          User Commands                          MKDIR(1)

NAME
       mkdir - make directories

SYNOPSIS
       mkdir [OPTION]... DIRECTORY...

DESCRIPTION
       Create the DIRECTORY(ies), if they do not already exist.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -m, --mode=MODE
              set file mode (as in chmod), not a=rwx - umask

       -p, --parents
              no error if existing, make parent directories as needed

       -v, --verbose
              print a message for each created directory

 Manual page mkdir(1) line 1 (press h for help or q to quit)
```

108
Dance with linux

5- استفاده از info

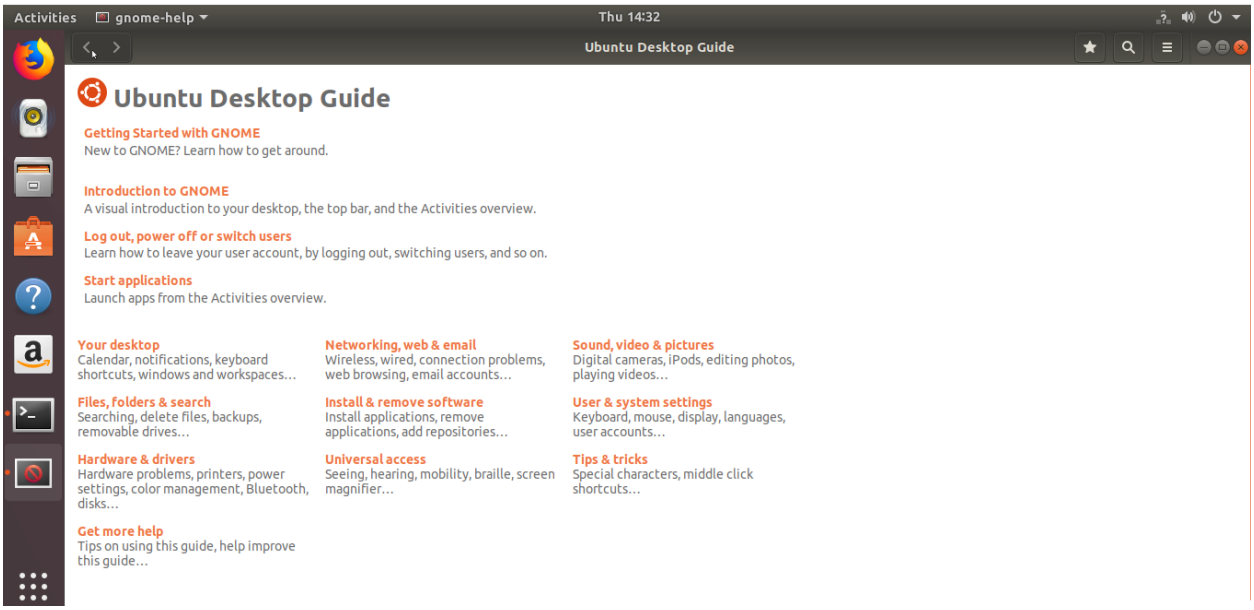
Info هم یکی از دستورات دیگری است که به دادن اطلاعات یا information به شما کمک می‌کند. منتهی تمام کامندها در دیتابیس info ندارند. اما اگر داشته باشند، اطلاعات بیشتری نسبت به man می‌دهند.

مثلا اینکه دیروز هم چی پوشیدن

تو نمی‌خوای پس کنی؟

6- استفاده از دستور gnome-help

مواردی که پیش از این گفتیم، در shell اجرا می‌شد اما با تایپ دستور gnome-help اپلیکشن help در محیط گرافیکی باز می‌شود که از آن هم می‌توانید استفاده کنید.



آرگمان‌ها (arguments)

خیلی از کامندها می‌توانند جدا از آپشن‌ها، آرگمان نیز بپذیرند.

ملت چه توانایی‌هایی دارن. نمی‌شه منم به فرزند خوندگی قبول کنن؟

یک آرگمان، یک بخش اضافی از اطلاعات به شمار می‌رود.

یعنی کلا اضافی‌ان؟

اطلاعاتی مثل اسم یک فایل، دایرکتوری، username و آیتم‌های دیگری که از کامند درخواست می‌کند آن‌ها را نشان دهد.

عین پلیسا رفتار می کنه. جاسوسه؟

مثلا دستور cat /etc/passwd محتویات فایل /etc/passwd را نشان می‌دهد. در اینجا /etc/passwd آرگمان به شمار می‌رود.

```
sahar@dance-with-linux:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologi
```

جایی که آرگمان‌ها با آپشن‌ها قاطی می‌شوند

آپشن‌ها در واقعی نوعی آرگمان‌اند.

گاهی اوقات یک آرگمان با یک آپشن شریک می‌شود (!) در این حالت، آرگمان فورا بعد از آپشن قرار می‌گیرد.

وقتی می‌خواهیم به دنبال آپشن‌های تک حرفی، آرگمانی بیاوریم، بعد از یک space می‌آید. اما اگر آپشن مد نظر، تک حرفی نبود، برای اینکه آرگمان را به دنبال آن بیاوریم، از علامت تساوی = استفاده می‌کنیم.

مثلا: دستور ls --hide=Desktop

```
sahar@dance-with-linux:~$ ls --hide=Desktop
Documents  examples.desktop  Pictures  Templates
Downloads  Music             Public    Videos
sahar@dance-with-linux:~$
```

hide-- آپشنی است که روی کامند ls نشسته، از طرفی می‌دانیم که کامند ls کارش لیست کردن محتویات فایل یا دایرکتوری‌ای است که به آن می‌سپارند، hide--به کامند ls می‌گوید که موقع لیست کردن محتویات، دایرکتوری‌ها و یا فایل‌هایی که اسم آن‌ها Desktop است را نمایش نده.

چه خبیث. مگه اون تو چخبره؟

فقط نکته اینجاست که بین دو طرف تساوی هیچ space یا فاصله‌ای نیست. یعنی اول آپشن اومده بعد هیچ spaceای نیست و تساوی اومده، دوباره هیچ spaceای نیست و آرگمان آمده.

کشتی خودتو فهمیدیم اسپیس نداره دیگه

مثال زیر، یک آپشن تک‌حرفی است که به دنبال آن یک آرگمان آمده است:

$ tar -cvf backup.tar /home/sahar

```
sahar@dance-with-linux:~$ tar -cvf backup.tar /home/sahar
tar: Removing leading `/' from member names
/home/sahar/
/home/sahar/Downloads/
/home/sahar/Documents/
/home/sahar/.config/
/home/sahar/.config/dconf/
/home/sahar/.config/dconf/user
/home/sahar/.config/yelp/
/home/sahar/.config/pulse/
/home/sahar/.config/pulse/45babca58779465584607 39b7500ae5e-stream-volumes.tdb
/home/sahar/.config/pulse/45babca58779465584607 39b7500ae5e-device-volumes.tdb
/home/sahar/.config/pulse/45babca58779465584607 39b7500ae5e-default-source
/home/sahar/.config/pulse/45babca58779465584607 39b7500ae5e-card-database.tdb
/home/sahar/.config/pulse/45babca58779465584607 39b7500ae5e-default-sink
/home/sahar/.config/pulse/cookie
```

توی اینجا آپشن cvf– می‌گوید یک فایل به اسم backup با پسوند tar بساز و همه محتویات داخل دایرکتوری /home/sahar را داخل آن بگذار.

اونجا که می‌گه بعد از منظورش create که c مخففش اومده توی آپشن. اونجایی هم که می‌گه "فایل" منظورش file که توی آپشن مخففش f اومده. اون حرف v هم که وسط اومده، منظورش verbos (به معنی طولانی، دراز) که یه پیام متنی می‌ده که بگه من یک آپی که می‌خواستی ساختم. حالا اگر برید توی دایرکتوری /home/user یوشه هه رو می‌بینید.

متاکاراکترها

یکسری کاراکترهای خاص وجود دارند که نمی‌توانید آن‌ها را در کامندها استفاده کنید. به این‌ها متاکاراکتر گفته می‌شود.

هر کدام از این متاکاراکترها برای bash معنی خاصی دارد. انواع مختلفی هم دارند. یعنی از طریق این کاراکترها ما می‌تونیم با bash راحت‌تر حرف بزنیم و چیزهای بیشتری ازش بخواهیم. در ادامه به معرفی بعضی از این متاکاراکترها می‌پردازیم.

متاکاراکتر نقطه‌ویرگول

نقطه‌ویرگول یا semicolon یا ; کاراکتری است که دو دستور را از هم جدا می‌کند. اگر بنویسم:

date ; date ; date

سه تا خروجی جدا در bash می‌بینید.

```
sahar@dance-with-linux:~$ date;date;date
Tue Jun  5 09:46:46 MDT 2018
Tue Jun  5 09:46:46 MDT 2018
Tue Jun  5 09:46:46 MDT 2018
```

یعنی نقطه‌ویرگول نمی‌ذاره بقیه به عنوان آرگمان پاس داده بشن، هر کدوم رو یه کامند جدا می‌دونه

چه فهمیده. کار اینشون رو دوست دارم. بیاد من و لینوکس رو هم به هم وصل کنه، مثلا بزنه linux ; khare

اشتباه فهمیدی، کار نقطه‌ویرگول جدا کردن دوتا کامند از همدیگس نه وصل کردنش. یعنی می‌شه توی یه خط چندین کامند رو نوشت و با نقطه‌ویرگول جدا کرد.

لینوکس هم برای من معنی خاصی می‌ده، یه چیزی بیشتر از یه سیستم‌عامل

منم همینطور

متاکاراکتر space

Space یا فاصله، خودش یکی از متاکاراکترها به شمار می‌رود که کامند را از آرگمان جدا می‌کند. مثلا وقتی تایپ می‌کنیم ls –la بین کامند ls و آپشن la– فاصله گذاشتیم.

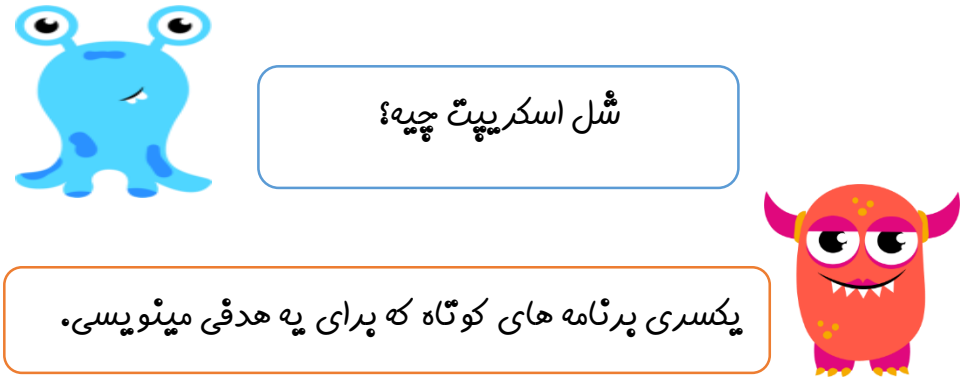
متاکاراکتر امپرسند &

بعضی از کامندها برای اینکه کامل اجرا شوند، طول می‌کشد (ماهم نسل بی‌حوصله) برای اینکه منتظر آن نمانیم، با کاراکتر امپرسند می‌توانیم دستور بعدی را در بک‌گراند run کنیم. به عبارتی این متاکاراکتر به شما اجازه می‌دهد تا به کارتان ادامه دهید و منتظر تکمیل کامند نمانید.

امپرسند معمولا در پایان یک برنامه می‌آید.

کامندهایی که مربوط به فرمت متن می‌باشند یا به اصطلاح text formatting commands مثل nroff یا troff که بعدا توضیح می‌دهیم، از این دست کامندهایی هستند که در background اجرا می‌شوند.

حتی ممکن است شما shell script هایی نوشته باشید که در background اجرا می‌شوند و یک اتفاقی که مدام در اتفاق افتادن است را چک می‌کند. مثلا میزان پرشدن هارد دیسک شما یا کاربران خاصی که لاگین کردند.



مثلا فرض کنید سه تا shell script با پایتون نوشته‌اید، برای استفاده از امپرسند اینگونه تایپ می‌کنید:

./script.py & ./script2.py & ./script3.py &

توجه کنید که & در پایان هر کدام آمده است.

متاکاراکتر دبل امپرسند &&

این کاراکتر جدا کننده دوتا کامند است.

Date && cal

```
sahar@dance-with-linux:~$ date && cal
Tue Jun  5 09:51:20 MDT 2018
     June 2018
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

sahar@dance-with-linux:~$
```

عادت داری حرف تکراری بزنی؛ فرقش با نقطه‌ویرگول چیه؛

هر کامندی وقتی کارش تمام می‌شود، یک status code به bash می‌فرستد تا وضعیت حال خودش را گزارش دهد. *یعنی بگه من کامندو به خوبی اجرا کردم یا قاطی کردم و مشکلی پیش اومده*

مثلا اگر صفر بده، یعنی برنامه رو اجرا کردم. عملیات با موفقیت انجام شد!

توی اینجا، وقتی bash متوجه شود که status code اولی، یعنی date صفر بوده، یعنی با موفقیت انجام شده، دومی یعنی cal را اجرا می‌کند.

اگر کامند اول حالش خوب نباشد و درست اجرا نشود، سراغ دومی هم نمی‌رود. به عبارتی با استفاده از دابل امپرسند، ما شرط می‌گذاریم که اگر اولین کامند اجرا نشد بقیه را هم اجرا نکن.

متاکاراکتر پایپ، pipe با علامت |

متاکاراکتر پایپ یک خط عمودی است، معمولا در کیبوردها با گرفتن شیف+ بک‌اسلش ظاهر می‌شود. این متاکاراکتر، خروجی حاصل از یک دستور را به عنوان ورودی دیگری ارسال می‌کند.

یعنی از اولی می گیره به دومی می‌ده

با این‌کار فرصت این را داریم که دستور اول روی برخی داده‌ها کار کرده، و دستور دوم روی خروجی دستور اول کار کند.

نفهمیدم

فکر کن خروجی دستور اول یه چیزی می‌شه مثل X، دستور دوم روی این خروجی یعنی X کار می‌کنه. انگار یکی قبلا پخته این فقط برمی‌داره می‌خوره و کارهایی که بهش سپردن رو روی اون انجام می‌ده.

الگویی شبیه به این دارد:

Command1 | command2

```
sahar@dance-with-linux:~$ ls /usr/bin |sort |less
sahar@dance-with-linux:~$
```

این دستور، محتویات دایرکتوری /usr/bin را لیست کرده، محتویات را بر اساس حروف الفبا (بدون در نظرگرفتن کوچکی و بزرگی حروف) مرتب می‌نماید و خروجی را به دستور less ارسال می‌کند. با این کار شما می‌توانید نتایج را به صورت صفحه به صفحه ببینید. برای خروج هم کلید q را فشار دهید.

یک مثال دیگر:

```
sahar@dance-with-linux:~$ cat /etc/passwd |sort |less
sahar@dance-with-linux:~$
```

این دستور محتویات فایل /etc/passdw را لیست می‌کند و خروجی آن را به دستور sort وصل می‌کند، دستور sort اسامی کاربرانی که در هر فایل /etc/passwd شروع می‌شوند را گرفته و آن‌ها را برحسب حروف الفبا مرتب می‌کند و در آخر خروجی آن‌را به less ارسال می‌کند. دستور less هم خروجی را به صورت صفحه به صفحه نمایش می‌دهد.

پایپ‌ها به خوبی این حقیقت را نشان می‌دهند که یونیکس همان جد لینوکس (!) چجوری از بلوک‌های مختلف ساخته شده‌اند.

متاکاراکتر دابل پایپ‌لاین

date || cal

```
sahar@dance-with-linux:~$ date || cal
Tue Jun  5 09:55:25 MDT 2018
sahar@dance-with-linux:~$ date2 || cal
No command 'date2' found, did you mean:
 Command 'date' from package 'coreutils' (main)
date2: command not found
     June 2018
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
```

در این متاکاراکتر اگر date درست اجرا شود یعنی status code آن صفر باشد، دومی دیگر اجرا نخواهد شد. به عبارتی این متاکاراکتر، هر کدام که صحیح بود را اجرا می‌کند. "این یا آن"

اگر هر دو صحیح بود، اولی را اجرا می‌کند، در غیر این صورت سراغ دومی می‌رود و الی آخر.

به عبارتی این شرط را می‌گذاریم که هر کدام از کامندها که حالش خوب بود آن‌را اجرا کن. حتی یکی! فقط انجامش بده.

همانطور که در تصویر می‌بینید، کامند date2 اجرا نمی‌شود پس سراغ کامند بعدی رفته و cal را نمایش داده است.

متاکاراکتر < یا خروجی استاندارد

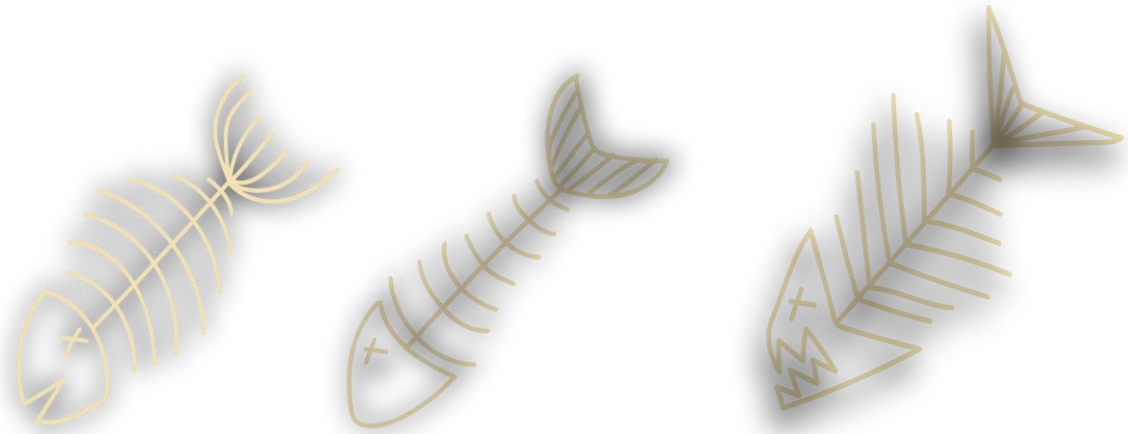
برگردیم سر bash

وقتی یک کامندلاین تایپ می‌شود تا bash آن‌را اجرا کند، bash برنامه شما را اجرا کرده یا به عبارتی execute می‌کند و بعد، آرگمان‌ها را به برنامه پاس می‌دهد. حالا برنامه ممکن است از آن‌ها استفاده بکند یا نکند. اینجا پراسس یا proccess ایجاد می‌شود. یک پراسس در طول عمر خود، هر وقت که تمایل داشت می‌تواند خروجی ارسال کند. مثل زمان که در برنامه پایتون با دستور print می‌گفتیم خروجی را ارسال کن و مثلا روی صفحه یک سلامی به ما بده hello را نشان بده. مشابه همین، این خروجی به دست bash می‌رسد، bash به طور دیفالت این خروجی را روی صفحه نمایش، نشان می‌دهد، اما کاراکتری وجود دارد به اسم "بزرگتر از" یا < یا greater than به طور مخفف gt.

یادم نمی‌مونه، من همیشه < یا > قاطی می‌کنم. از کجا بفهمم؟

ببین دهن این ماهیه به کدوم طرفه؛ هر جا که غذا بیشتر باشه ماهی به اون سمت دهنش بازه.

ماهیه چرخید چی؛ اصن دهن ماهی‌های من ^ این شکلین.



عاشق اونیم که دهنش بستس. خره‌جان سختش نکن دیگه. یه شکلو یاد بگیر دیگه قاطی نکنی.

اصن منم می‌خوام یاد بگیرم.

منم بازی

نه تو دیگه. هر چی جک و جونوره اینجاست. یه اندازه کافی از خره می‌کشیم. تو یکی وارد نشو با اون حافظه سه ثانیه‌ایت

این علامت، یک متاکاراکتر به شمار می‌رود و هر وقت bash این کاراکتر را ببیند متوجه می‌شود که باید خروجی را در یک فایل بریزد. چه فایلی؟ فایلی که شما نام آن‌را بعد از علامت بزرگتر می‌آورید.

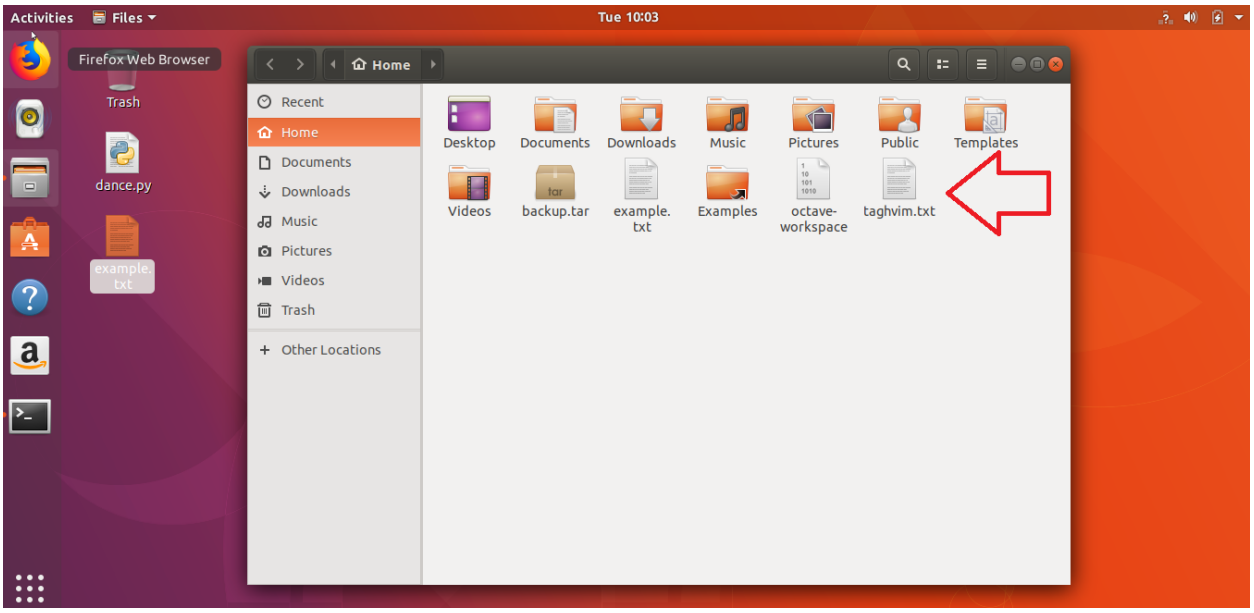
مثلا در ترمینال می‌نویسم: cal

با این کار تقویم را روی صفحه نمایش به من نشان می‌دهد، اما اگر بنویسم:

cal > taghvim.txt

```
sahar@dance-with-linux:~$ cal > taghvim.txt
sahar@dance-with-linux:~$
```

دیگه خروجی‌ای روی صفحه نمی‌بینید، و خط prompt برای گرفتن دستور بعدی مدام چشمک می‌زند و این یعنی دستور قبلی اجرا شده. اما خروجی کجا رفته؟ خروجی درون فایل taghvim.txt درون دایرکتوری جاری حضور دارد. دایرکتوری جاری ما الان فولدر /home/sahar است. (یادمون نره که با pwd هم می‌تونستیم ببینیم توی چه دایرکتوری هستیم)



نکته مهم اینجاست هر بار که این علامت بزرگتر را استفاده می‌کنیم، آن فایل مورد نظر (اینجا taghvim.txt) خالی می‌شود، پاک می‌شود و خروجی دستور جدید در آن قرار می‌گیرد. اصطلاحا به این کار می‌گوییم overwrite شدن.

اگر بخواهیم overwrite نشود و خروجی به انتهای فایل اضافه شود، به جای یک بزرگتر، از دوتا علامت بزرگتر استفاده می‌کنیم.

فرض کنید می‌خواهیم date هم به آخر فایل تقویم اضافه شود:

date >> taghvim.txt

```
sahar@dance-with-linux:~$ cal > taghvim.txt
sahar@dance-with-linux:~$ date >> taghvim.txt
sahar@dance-with-linux:~$
```

Open ▾ | taghvim.txt ~/ | Save

```
     June 2018
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

Tue Jun  5 10:11:51 MDT 2018
```

اصطلاحا به این کار output redirection گفته می‌شود.

متاکاراکتر کوچکتر یا < یا ورودی استاندارد

قبلا هم اشاره کردیم که بسته به برنامه‌نویس یک برنامه، ممکن است آن برنامه خروجی داشته باشد یا نداشته باشد. این برای ورودی استاندارد نیز صدق می‌کند. یعنی با توجه به نظرات و تصمیمات (!) برنامه‌نویس محترم، ممکن است یک برنامه برای ادامه کارش از کاربر یا همان یوزر، ورودی درخواست کند.

مثل موقعی که به این یخچال گنده‌ها پول می‌ندازی توش تا بهت آب معدنی بده. تا پولو ندی ادامه نمی‌ده.

اگر برنامه‌ای داشته باشید که از ورودی یا input استفاده می‌کند، bash مستقیما کیبورد شما را به آن برنامه متصل می‌کند.

نقش دستمون رو داره موقع پول انداختن توی یخچاله. نه؟

یعنی از اینجا به بعد هر چی با کیبورد بنویسید و یک Enter بزنید، آن اطلاعات به عنوان ورودی یا input فرستاده می‌شود. دیگه با آن ورودی، برنامه چه کاری انجام می‌دهد به ما ربطی نداره.

چطور ربطی نداره؟ پاسخگو باش

مثلا برنامه یه جای خالی داده این شکلی: ----- = 2 *---- و از تو یه عددی که توی جای خالی اول می‌خوای قرار بگیره رو می‌پرسه. این مسئله فرقی توی نحوه خریت تو داره یا؟

اره. اصلا من دوست دارم برنامه جوری کار کنه که دو دوتا رو بزنه 5.7

هفت‌ونیم؟؟؟؟؟؟ چطور ممکنه؟

برنامه خودمه دوستت دارم اینجوری باشه. خیلی دخالت می‌کنیا. می‌گم از سناریو بندازنت بیرون. حریم شخصی حالیت نیست؟

برنامه تا وقتی که از یوزر ورودی می‌خواهد، متوقف می‌شود. برای مثال، در مثال دو دوتا می‌شه هفت و نیم، برنامه منتظر می‌ماند آن 2 یا هر عددی دیگر را وارد کنید تا ضرب انجام گیرد و خروجی نشان داده شود.

حتی می‌توان با bash صحبت کرد و گفت که اگر ورودی یا input احتیاج داشتی، بجای آنکه مستقیم به کیبور وصل شوی، این input را از یک فایل بگیر.

خوبه جدیدا با bash هم حرف می‌زنی! چشمم روشن. فشار امتحانا تته دیگه.

مثلا فرض کنید برنامه‌ای نوشتیم (با اسم danceWithLinux) و در محیط bash اجرا کردیم. و می‌دانیم که این برنامه از یوزر ورودی می‌خواهد. اگر بخواهیم بگوییم ورودی را از یک فایل (فایلی با اسم data) بگیر، باید بنویسیم:

```
danceWithLinux < data
```

در این حالت برنامه danceWithLinux ورودی خود را از data می‌گیرد.

برای درک بهتر، فکر کنید برنامه‌ای به زبان پایتون نوشتید که می‌خواهد معدل دانش‌آموزان یک کلاس را محاسبه کند. و حالا نمرات این دانش‌آموزان درون فایلی به اسم nomarat وجود دارد. برای آن که به برنامه بفهمانید که nomarat را به عنوان ورودی بپذیرد باید بنویسیم:

moadel < nomarat

یکم بازی کنیم؛ اگر این را بنویسم چه اتفاقی خواهد افتاد؟

moadel < nomarat > final.txt

عجب خر تو خری شد

پی ادب! در این حالت نمرات رو به عنوان ورودی به برنامه moadel می‌ده، بعد از اینکه معدل رو حساب کرد، خروجی رو توی final.txt وارد می کنه.

یعنی هم از ورودی استاندارد استفاده کردیم، هم از خروجی استاندارد. و مهم‌تر اینکه به شما در خودِ bash نه چیزی نشان خواهد داد، نه چیزی خواهد پرسید.

فقط اینکه اینجا فایل final.txt، به اصطلاح overwrite می‌شود. اگر نخواهیم این اتفاق بیافتد، دوتا علامت بزرگتر استفاده می‌کنیم:

moadel < nomarat >> final.txt

چه بچه خوبیه. سر به راهه. دوسش دارم. کار خودشو می کنه.

با هم یه مورد عینی را امتحان کنیم. در bash برنامه‌ای داریم به اسم wc که مخفف word count است. این برنامه از یوزر، ورودی می‌خواهد. برای همین متوقف می‌شود تا یوزر، ورودی‌های خودش را از طریق کیبورد وارد کند.

کاربر (که من و شما باشیم) برای اینکه به برنامه بفهمانیم ورودی‌هایی که می‌خواستیم را ارسال کردیم و تموم شد و خلاصه ولمون کن بابا و اجراش کن، باید آخرش ctr+d را بزنیم:

```
sahar@dance-with-linux:~$ wc
Dance with linux
sahar shaker
bug blog
```

اینجا باید ctr+d را بزنم:

```
sahar@dance-with-linux:~$ wc
Dance with linux
sahar shaker
bug blog
      3       7      39
sahar@dance-with-linux:~$
```

بهم می‌گه: سه خط نوشتی، هفتا کلمه بود، و 39تا حرف.

فان: دروغ می گه من می‌خوام بشینم بشمارمش.

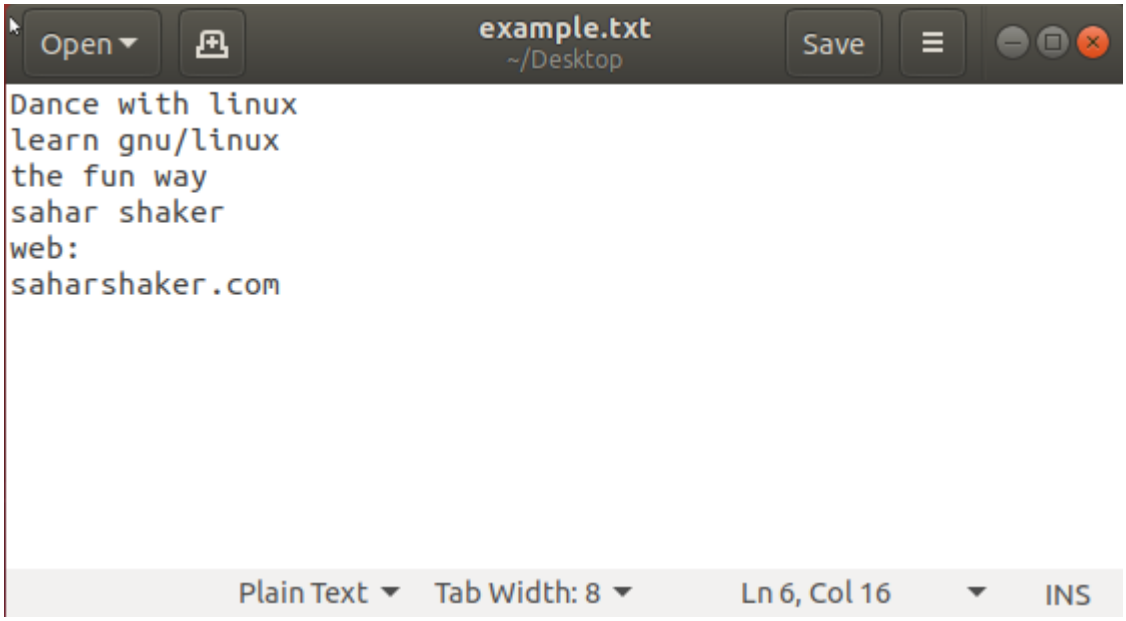
دست شما درد نکنه دیگه شدیم دروغگو

ای جانم! کجا بودین شما تا الان؟ خودش خالی بنده بابا. جلو خودم، بیستا کارت پستال "عزیزم تو تنها عشق منی" خرید. شما غم به دلت راه نده.

کامند wc هم مانند سایر کامندها، آپشن‌های زیادی دارد.

بازی خودمان را ادامه دهیم، به wc بگوییم ورودی را از یه فایل مثلا به اسم example بگیر و خروجی را در note.txt بریز:

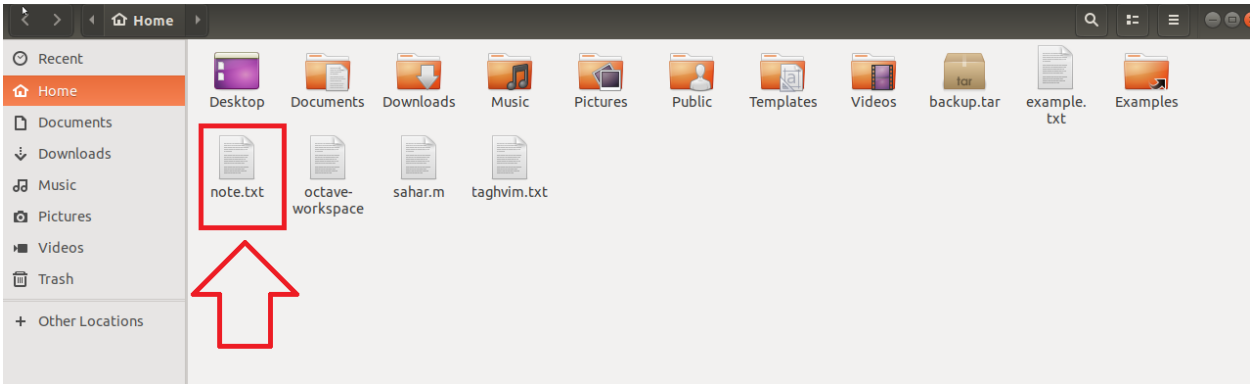
فرض کنید فایل example.txt محتویاتی این چنینی دارد:



می‌خواهیم:

```
sahar@dance-with-linux:~$ wc < example.txt >> note.txt
sahar@dance-with-linux:~$
```

به طور خودکار، فایلی به اسم note.txt در دایرکتوری جاری ایجاد می‌شود که نتیجه را در آن نوشته:



تا اینجا یاد گرفتیم که چگونه به bash بفهمانیم تا مسیر ورودی استاندارد و خروجی استاندارد را عوض کند.

نکته: ما با متاکاراکتر پایپ‌لاین نیز یاد گرفته بودیم که خروجی یک برنامه را به ورودی برنامه دیگر وصل کند. بالاتر بروید پیدایش می‌کنید :)

گفته بودیم پایپ‌لاین با علامت | ، خروجی برنامه اول را به ورودی برنامه دوم متصل می‌کند. مثلا در شکل زیر، با دستور ls محتویات داخل دایرکتوری /dev را لیست کردیم، حالا لیستی که به دست آمده را به عنوان ورودی کامند wc ارائه دادیم، wc هم نتیجه را نشان می‌دهد:

فان: گلاب به روتون می‌شه انقدر wc نگید

شارده قراضه حساسه

```
sahar@dance-with-linux:~$ ls /dev | wc
    191     191    1210
sahar@dance-with-linux:~$
```

با bash کارهای خفن‌تری (!) می‌توانید انجام دهید. اما همگی بستگی به کارکردن شما دارد. با مدام خواندن هیچ کتابی و سایتی برنامه‌نویس نخواهید شد. کار و کار و کار.

بله از قدیما گفتن کار جوهر مرده

هر چی شما بگید

این چش شد؟ چرا کمرنگ؟

فکر کنم جوهرش تموم شد. آخه پیکارتر از این موجود وجود نداره!

در واقع کار پایپ‌لاین شبیه زنجیری است که کامندها را به هم وصل می‌کند.

متغیرهای محیطی (enviroment variables)

کامندها به طور کلی به دو دسته تقسیم می‌شوند:

1- کامندهای داخلی یا internal
2- کامندهای خارجی یا external

کامندهای داخلی به دستوراتی گفته می‌شود که کامپایل شده‌اند و درون خود bash وجود دارند. اما تعداد آن‌ها زیاد نیست. ممکن است کامندی را تایپ کنید که داخلی نباشد در این صورت bash به دنبال فایل اجرایی آن دستور می‌رود.

یعنی عملا می‌فرستیم دنبال نخود سیاه؟

بنابراین اگر کامندی داخلی نباشد، باید یک فایل یا برنامهٔ کامپایل شده‌ای باشد که shell آن را پیدا کند و توانایی خواندنش را داشته باشد.

کد کامپایل شده یعنی اونو توی زبان مثلا توی python نوشتیم و اون جویده و فایلش رو می‌ده، دیگه shell می‌تونه اونو بخونه

اینجا مهم است که shell در کدام فولدرها به دنبال این فایل‌های کامپایل شده می‌گردد.

مثلا فرض کنید یک برنامه‌ای با زبان پایتون نوشته‌ایم به اسم dance که درون آن خیلی ساده گفته‌ایم print('dance with linux') حالا کاری که باید انجام دهیم این است که کد کامپایل شدهٔ این برنامه dance را جایی قرار دهیم که bash بتواند شناسایی کند. بعد از اینکه درون اتاقی (پوشه‌ای، فایلی) مناسب قرار دادیم، می‌توانیم توقع داشته باشیم که با تایپ دستور dance و اینتر زدن داخل bash، دستور اجرا شده و عبارت dance with linux روی اسکرین ظاهر شود.

اسم این فایل اجرایی باید با کلمهٔ اول کامند یکی باشد.

اینجاست که چیزی به اسم متغیر محیطی مطرح می‌شود. مثل خیلی از زبان‌ها که اجازه داریم درون آن‌ها چیزی به اسم متغیر تعریف کنیم، چنین امکانی در bash نیز وجود دارد که این متغیر در موری ذخیره می‌شود.

در واقع bash از متغیر محیطی path لیست یکسری دایرکتوری را می‌خواند؛ دایرکتوری‌ها را به ترتیب و یکی‌یکی می‌گردد و اگر فایل باینری که اسمش با کلمهٔ اول کامند شما یکی بود پیدا کرد آن‌را اجرا می‌کند و بقیهٔ کامند شما را که با space از هم جدا شده‌اند را به صورت آرگمان اجرا می‌کند.

من یه چیز می‌دونم، اینجا بهش می‌گن آرگمان رو به اون پراسسی که داره اجراش می‌کنه، پاس می‌ده

بوگو بینیم کی گل می‌شه؟

گلش پستگی به خودت داره و زاویه شوتت! تو هم که کلا شوتی مطمئن باش روی تو یکی حواب می‌ده

مثلا کامند ping یک کامند اکسترنال است و bash آن‌را اجرا می‌کند. پس bash توانسته فایل اجرایی ping را پیدا کند.

```
sahar@dance-with-linux:~$ ping
Usage: ping [-aAbBdDfhLnOqrRUvV64] [-c count] [-i interval] [-I interface]
            [-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
            [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
            [-w deadline] [-W timeout] [hop1 ...] destination
Usage: ping -6 [-aAbBdDfhLnOqrRUvV] [-c count] [-i interval] [-I interface]
            [-l preload] [-m mark] [-M pmtudisc_option]
            [-N nodeinfo_option] [-p pattern] [-Q tclass] [-s packetsize]
            [-S sndbuf] [-t ttl] [-T timestamp_option] [-w deadline]
            [-W timeout] destination
```

در مورد چگونگی نوشتن متغیرهای محیطی و تنظیم آن‌ها در فرصتی جدا بحث خواهیم کرد.

یعنی دستی دستی پیچوندی مارو سحر؟

نه نوشتن متغیرهای محیطی خودش به برنامه‌نویسی نیاز داره و چندتا کامند دیگه.

باشه ولی من خود سیاهمو همین‌جا گم کردم. خواستم بگم حواست هست.

کامندهای پایه‌ای

پیش از این، در بین مثال‌ها به چندین کامند پایه‌ای اشاره شد. مثل ls

در این‌جا سعی می‌کنیم به چندین کامند دیگر اشاره کنیم. تمامی این کامندها حکم جعبه‌ابزاری را دارند که باعث می‌شوند کارهایتان را سریعتر و راحت‌تر انجام دهید.

قبل از اینکه بحث را شروع کنیم، خوب است اشاره کنیم که در بخش "پیداکردن اطلاعات در مورد کامندها" به خوبی توضیح دادیم چگونه می‌توانیم کارکرد یک کامند را بفهمیم و همچنین آپشن‌ها و آرگمان‌هایی که هر کامند می‌تواند بگیرد را چگونه استخراج کنیم. برای همین خوب است نیم نگاهی به آن بخش کنید.

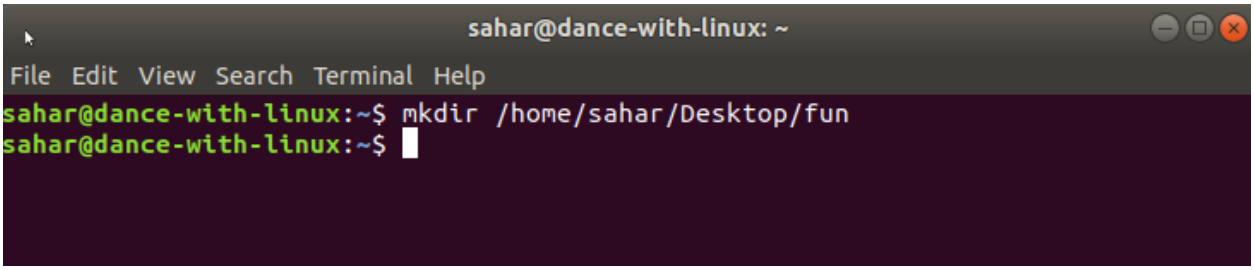
کامند mkdir

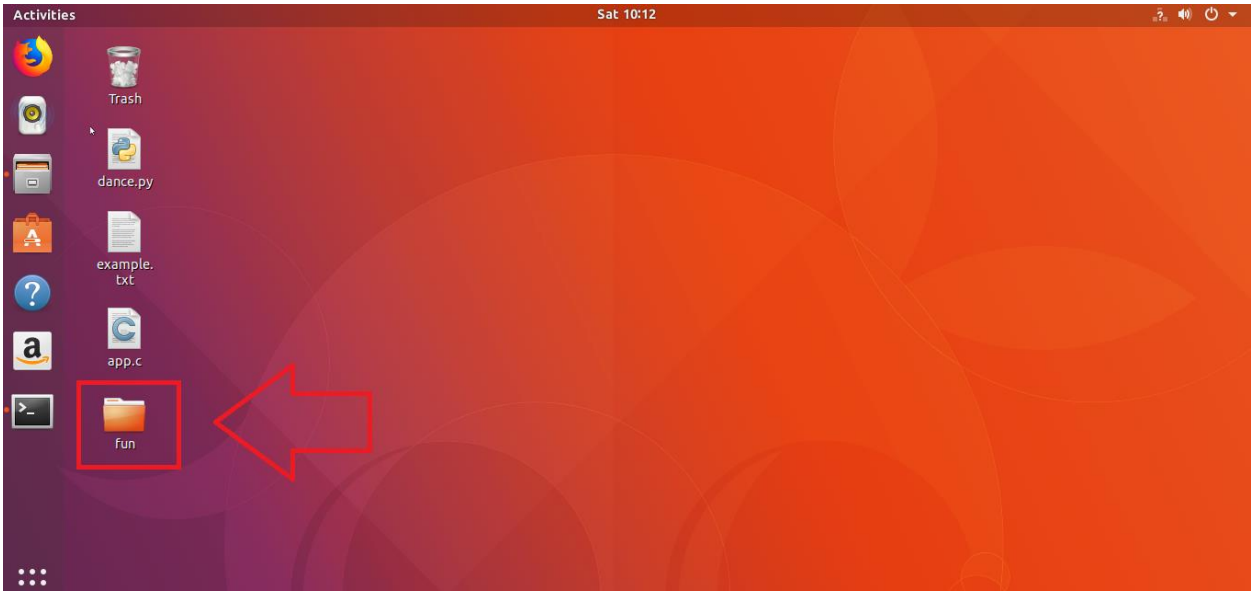
با این کامند می‌توان فولدر یا دایرکتوری ساخت.

یعنی خونه دار میشیم؟

اول باید به این دستور، یک دایرکتوری بگیم، این دایرکتوری باید وجود داشته باشد. مثلا می‌توانیم بنویسیم:

```
mkdir /home/sahar/Desktop/fun
```





یعنی طرف توقع داره دیگه حضرت والا، تا قسمت desktop رو داشته باشین و ایشون فقط بیاد اتاق رو فرشش کنه. یعنی اون قسمت fun رو خودش می‌سازه.

از کجا بفهمیم که تشکیل شده است؟

دستور ls محتویات داخل دایرکتوری جاری را لیست می‌کند. اگر در دایرکتوری جاری حضور خود را رسانده باشیم، می‌توان با bash صحبت کرد و راضی به نشان دادنش کرد.

یعنی بنویسیم:

ls –l

```
sahar@dance-with-linux: ~/Desktop
File  Edit  View  Search  Terminal  Help
sahar@dance-with-linux:~$ mkdir /home/sahar/Desktop/fun
sahar@dance-with-linux:~$ cd /home/sahar/Desktop
sahar@dance-with-linux:~/Desktop$ ls -l
total 16
-rw-rw-r-- 1 sahar sahar  174 Jul  6 07:34 app.c
-rw-rw-r-- 1 sahar sahar   27 May 11 05:45 dance.py
-rw-rw-r-- 1 sahar sahar   82 Jun  5 07:58 example.txt
drwxr-xr-x 2 sahar sahar 4096 Aug 11 10:10 fun
sahar@dance-with-linux:~/Desktop$
```

به محض لیست گرفتن، می‌بینید که فولدری به اسم fun ساخته شده است.

اگر بجای sahar توی دایرکتوری بنویسم ashpazkhooneچی می‌شه؟شاید دلم بخواد بره آشپزخونه

هیچی. ارور می‌ده. چون توقع داره حداقل وسعت بکشه و تا قسمت آشپزخونه رو تو داشته باشی که نداری. یعنی یه مسیر، یه اتاق درست باید بهش بدی تا بتونه اونجا زندگی کنه

```
sahar@dance-with-linux: ~
File  Edit  View  Search  Terminal  Help
sahar@dance-with-linux:~$ mkdir /home/ashpazkhoone
mkdir: cannot create directory ‘/home/ashpazkhoone’: Permission denied
sahar@dance-with-linux:~$
```

سرویس چی؟ اونو دارم

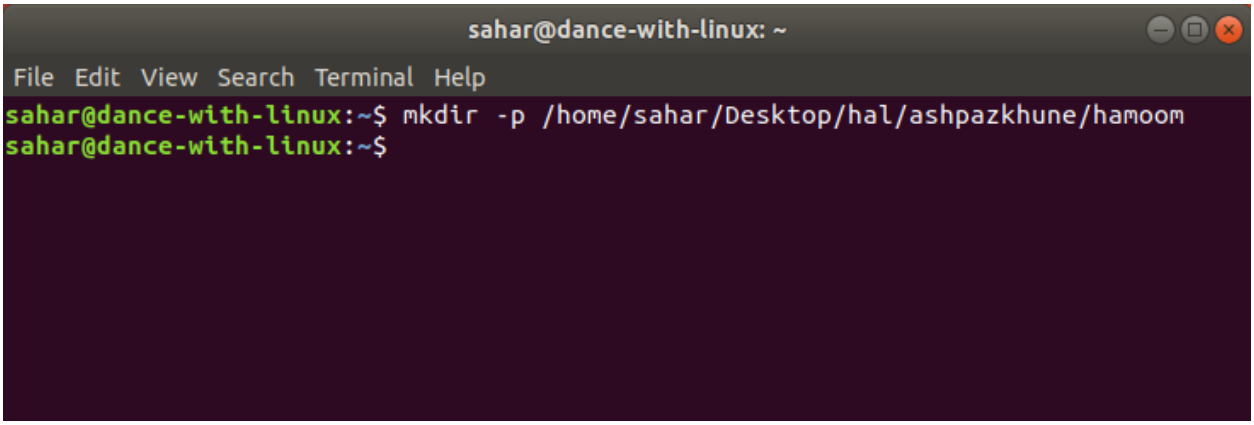
آپشن p در کامندmkdir

این دستور، یک آپشنی دارد به اسم p- که می‌توان با استفاده از آن، دایرکتوری‌های تو در تو ساخت.

یعنی توی حال، آشپزخونه بسازیم، توی آشپزخونه، حموم بسازیم و برو تا آخر

مثلا:

mkdir –p /home/sahar/Desktop/hal/ashpazkhune/hamoom



حالا اگر از دایرکتوری، لیست بگیریم:

ls

نشان می‌دهد که اول حال را ساختیم، بعد داخل حال، آشپزخونه ساختیم و داخل آشپزخونه حموم ساختیم.

یعنی اگر از حال، lsبگیریم:

ls hal

آشپزخونه را نشان می‌دهد

اگر از آشپزخونه ls بگیریم:

ls hal/ashpazkhune

حموم را نشان می‌دهد.

```
sahar@dance-with-linux: ~/Desktop
File  Edit  View  Search  Terminal  Help
sahar@dance-with-linux:~/Desktop$ ls
hal
sahar@dance-with-linux:~/Desktop$ ls hal
ashpazkhune
sahar@dance-with-linux:~/Desktop$ ls hal/ashpazkhune
hamoom
sahar@dance-with-linux:~/Desktop$
```

نقش tree درmkdir

اگر tree را هم تایپ کنین (خالی تایپ کنین)، فایل‌های تو در تو را به صورت درختی نشان می‌دهد.

اینجوری خونه رو از نمای بالاتر میتونین ببینین، ویوتون هم عالی می‌شه. فقط مواظب باشین کسی دستشویی نباشه

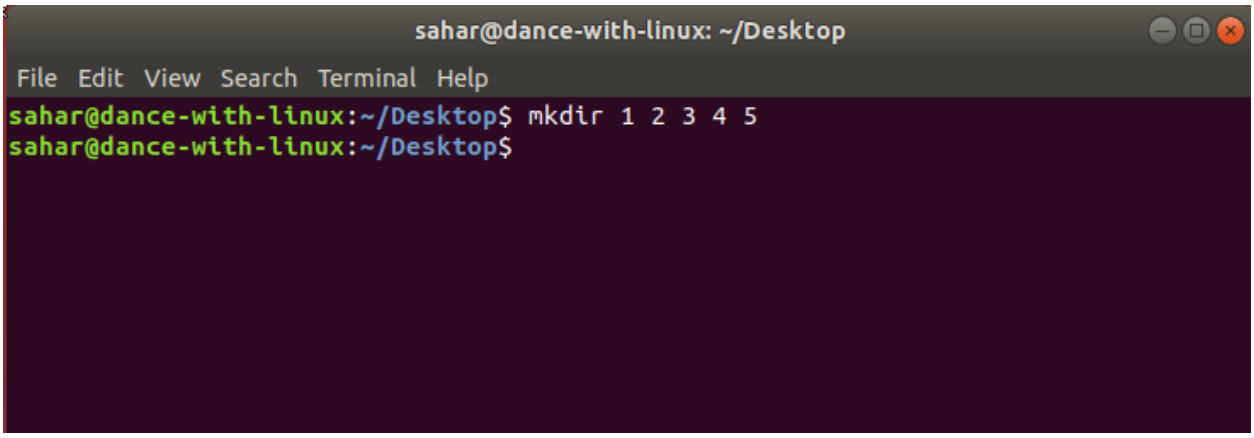
حالا نوبت شماست. چهارتا فولدر تو در تو بسازین ببینیم چقدر خونه سازی بلدین؛ مثلا اسم فولدر اول رو بذارید 1، بعد داخل یک، فولدر 2 رو بسازین، بعد داخل 2، فولدر 3 رو بسازین. جواب‌هارو می‌تونید توی این آدرس، کامنت کنین تا بقیه هم ببینن

راستی mkdir معلومه مخفف make directory است.

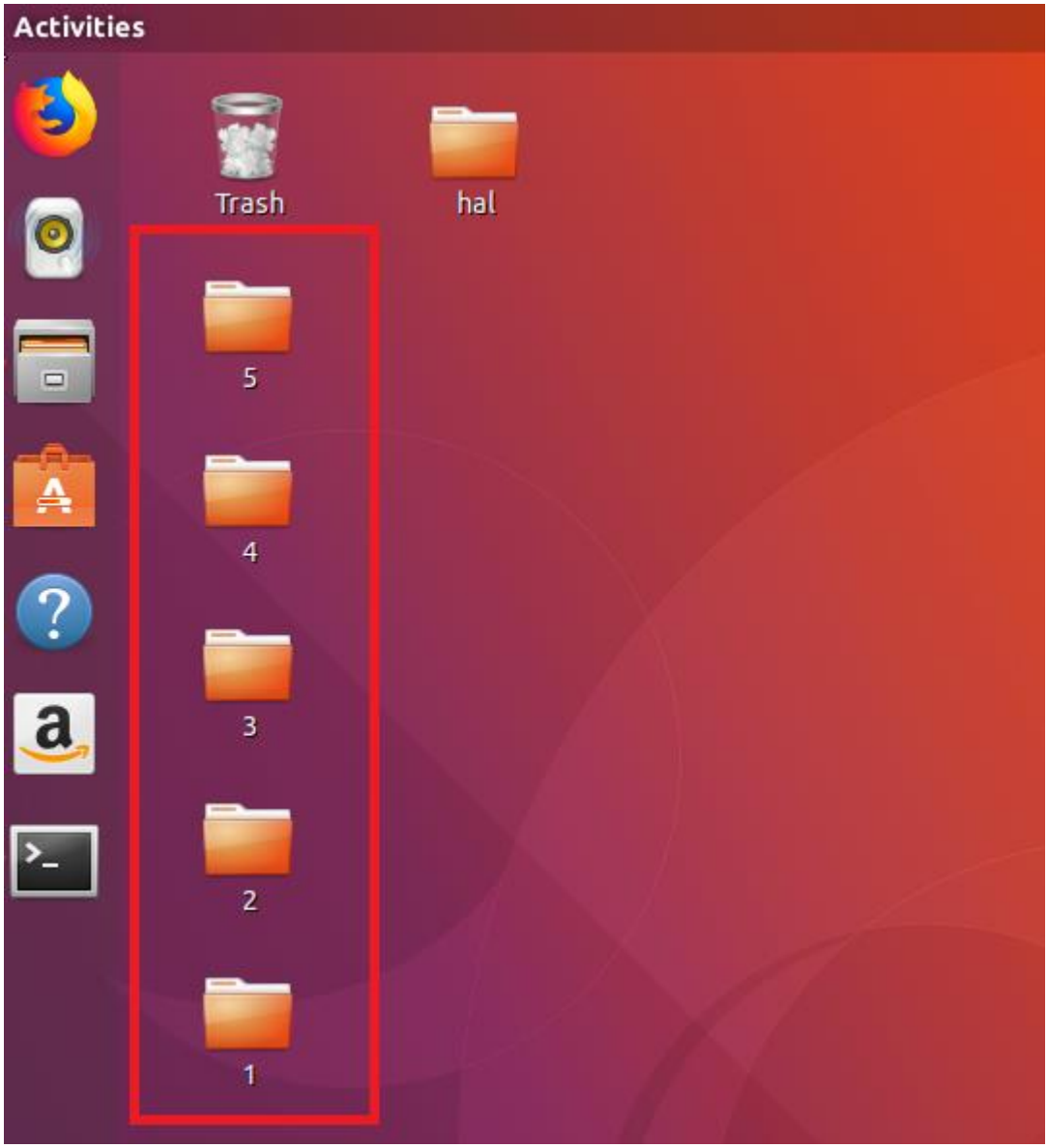
چطوری چندتا دایرکتوری رو باهم توی یک خط کامند بسازیم؟

برای این کار، کافی است بین اسامی، فاصله یا space قرار دهید:

mkdir 1 2 3 4 5



در این صورت 5 فولدر، در **دایرکتوری جاری** شما (اینجا دسکتاپ) ایجاد می‌شود.



با چی می‌فهمیدیم توی کدوم اتاقیم یا همون
دایرکتوری جاری رو از کجا پیدا می‌کردیم؟

با تایپ pwd

چرا هر چی می‌نویسم، صدایی ازش درنمیاد؛
یعنی چرا هیچ چیزی روی صفحه نشون نمی‌ده؛

این کامند، مانند بسیاری از کامندهای دیگر، تا زمانی که به error و مشکلی نخورد، صدایش در نمی‌آید. آهسته کار خود را می‌کند و دستوری که به آن داده‌اید را به خوبی گوش می‌دهد.

در این مثال، کافی است به دایرکتوری جاری خود، مثلا اینجا desktop بروید تا ببینید که 5 فولدر به اسم‌های 1 و 2و 3و 4و 5 ساخته شده است.

اصلا می‌شه توی bash موند از دایرکتوری جاری، lsگرفت تا بفهمیم که کارشو کرده یا نه.

خب اینجوری اذیتت می‌شم، یکار دیگه کن که من خروجی رو ببینم.

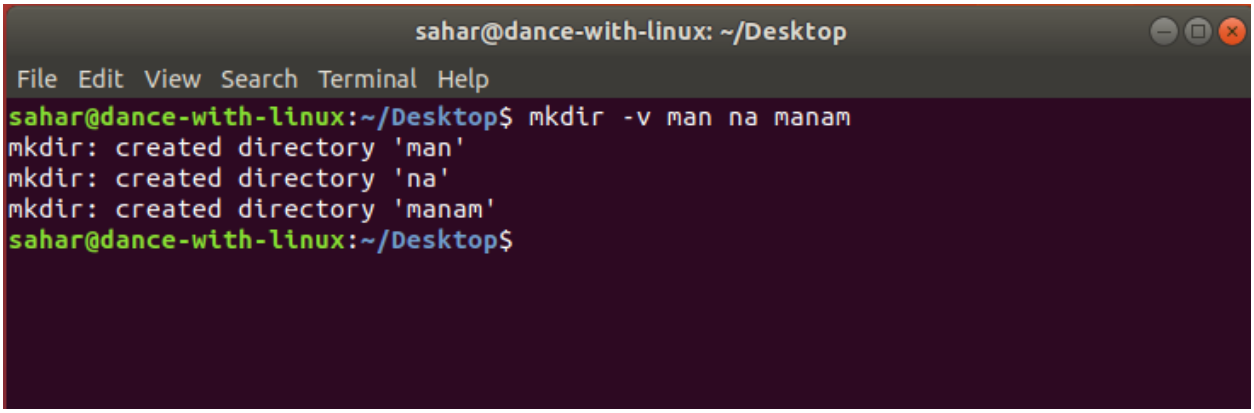
نمی‌شه زد تو گوشش تا صداش دربیاد و اعتراف کنه چه کارایی کرده؟

چه خشن

با اضافه‌کردن آپشن v– که مخفف verbose به معنای پرحرف است، می‌توان خروجی را در همان صفحه مشاهده کرد.

یعنی در این حالت، به شما یک پیام نشان می‌دهد که من ساختم.

mkdir –v man na manam



همان‌طور که در تصویر می‌بینید به شما اطلاع داده است که سه فولدر به اسم "من نه منم" ساخته است.

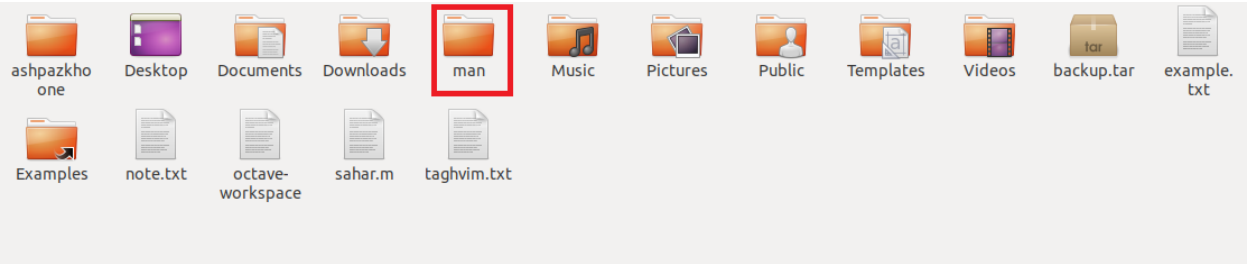
پاک‌کردن فایل، کامند rmdir

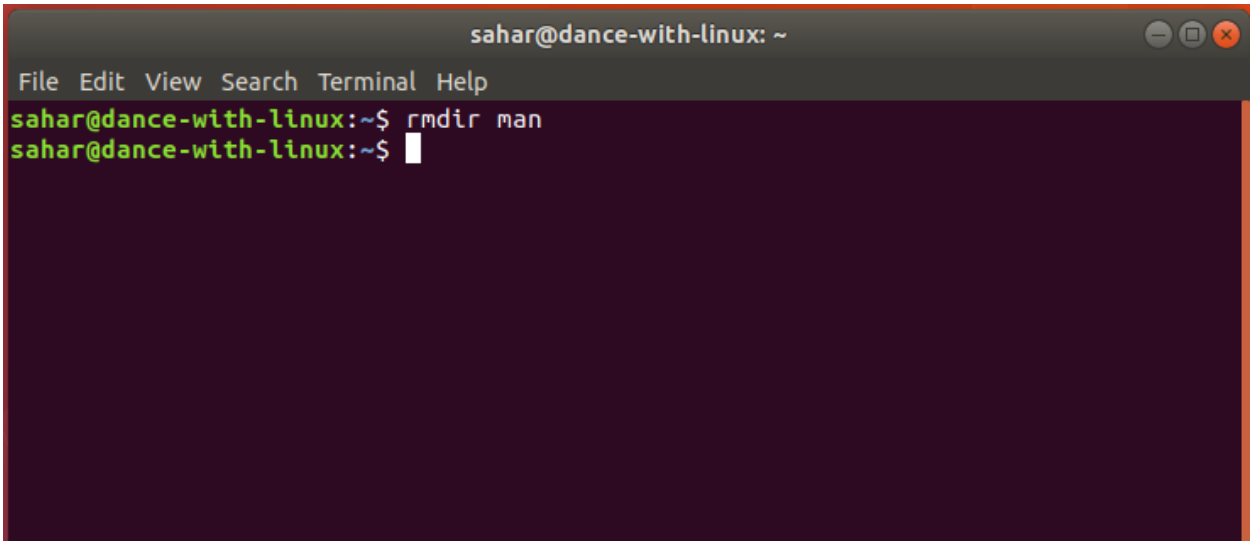
برای پاک‌کردن یک فایل، از کامند rmdir استفاده می‌کنیم.

اونْ rm اولشْ واسه remove هستْ؛

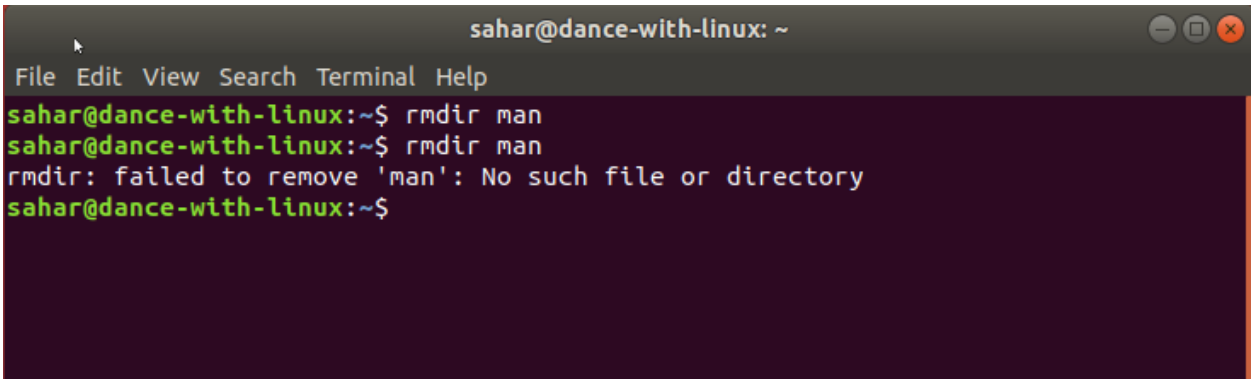
یاه! انقدر درکشْ سختْـــــــــــــه؛

مثلا برای پاک‌کردن فایل man می‌توان نوشت:



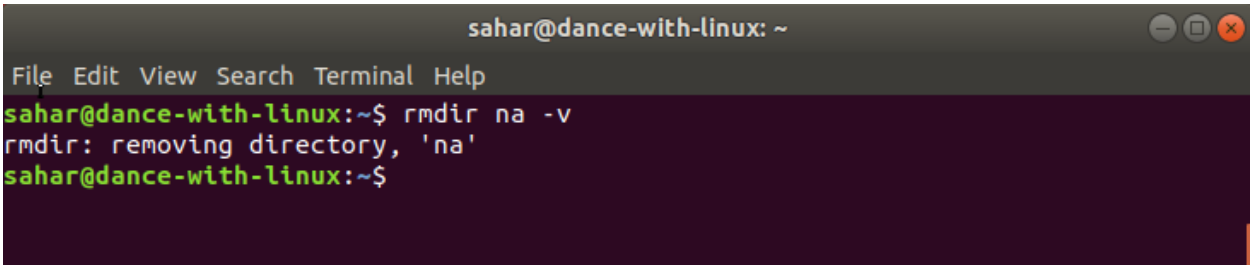


اگر دوباره بزنم:



ارور می‌ده که همچین فایلی کلا نیست که بخواهم پاکش کنم.

این کامند هم کامند ساکتی است برای به حرف درآوردن و اینکه یک پیغام نشان دهد که من حذف کردم، آپشن v- که قبلا هم گفته شد، کنارش می‌آید:



اگر از دایرکتوری جاری، ls بگیریم، می‌بینیم که پاک شده است.

تفاوت فایل با دایرکتوری

دایرکتوری چیزی نیست که روی هارد دیسک وجود داشته باشد و دارای contentای باشد؛ بنابراین پاک کردنش چندان هم معنا ندارد. پاک‌کردن دایرکتوری یعنی بیایم از درخت دایرکتوری‌های فایل سیستم حذف کنیم. از یک دیتابیسی حذفش کنیم. با این حال اگر یک دایرکتوری پُر باشد، نمی‌توان آن‌را پاک کرد.

از قبل می‌دانیم که برای دیدن دایرکتوری‌ها کافی از کامند tree استفاده کنیم.

اگر tree را تایپ کنیم:

در tree می‌بینیم که در دایرکتوری 1، دایرکتوری 2 وجود دارد، و در دایرکتوری 2، دایرکتوری 3 است و بعد هم دایرکتوری 4 در دل دایرکتوری 3 وجود دارد. پس اگر بزنم:

rmdir 1 ارور می‌دهد چرا که دایرکتوری 1، پُر است.

زورش نمی‌رسه rmdir یعنی می‌خوای بگی دایرکتوری پاک کنه ولی فایل رو می‌تونه پاک کنه؟

همینه. rmdir زورش به دایرکتوری نمی‌رسه و فقط می‌تونه از لیست tree حذف کنه با این حال نمی‌شه به دایرکتوری رو پاکش کرد.

اونوقت فیلسوف‌جان "پاک‌کردن" با "حذف‌کردن" چه تفاوتی داره؟

از لحاظ ادبیاتی نمی‌دونم ولی وقتی می‌گم پاک می‌کنه یعنی delete و یعنی دیگه نیست، دیگه وجود نداره. ولی وقتی می‌گم پاک‌کنه یعنی remove و یعنی فقط از لیست tree می‌شه دایرکتوری رو حذف کرد. ما فایل رو پاک می‌کنیم، ولی دایرکتوری رو تنها از لیست حذف می‌کنیم.

خب any way

این معلومه نفهمیده که انگلیسی گفت.

ساخت فایل

برای ساختن یک فایل خالی خیلی راه وجود دارد.

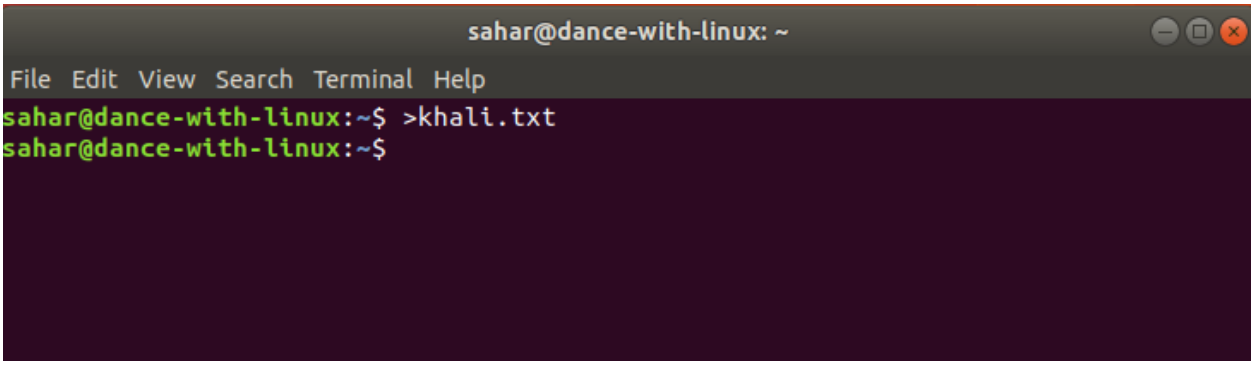
"خ" رو دوست ندارم می‌شه زیاد استفاده نکنی؟

کجا دیدی؟

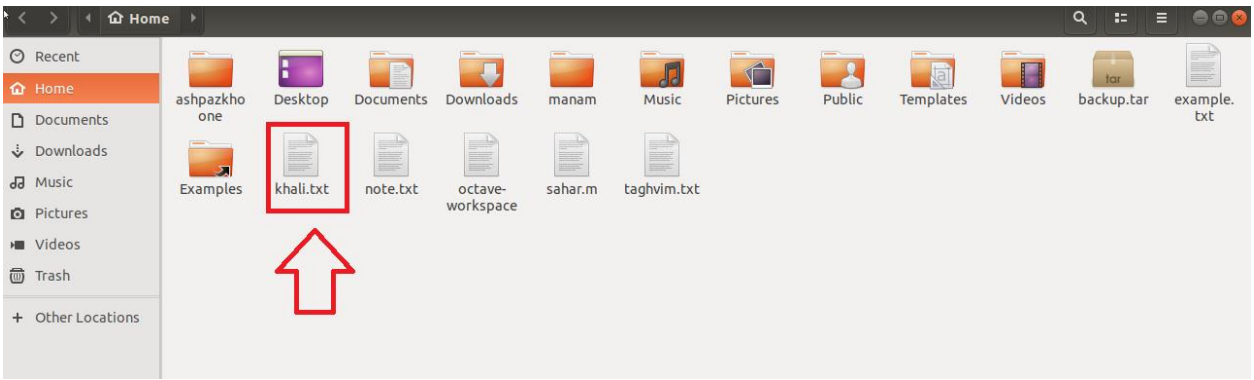
اینجا دیگه "خالی خیلی"

آخ‌جون که تو خوشت نیاد؛ راستی، خلیل توی خیابون زیر درخت خیار می‌خورد، خیلی هم خوشش می‌اومد. انقدر هم خندید، فکر کنم تا فیها خالدونت سوخت. اصلا خدا باید اسم حوا رو می‌ذاشت خوا. شاید اینجوری بشریت به طریق بهتری می‌رفت ما هم راحت‌تر بودیم. خلاصه که خعلی خوش گذشت. می‌خوام از این به بعد به هوا هم بگم خوا. خالت خوش رفیخ. دور هم می‌خوایم بشینیم و یه تجدید نظر توی واژگان بکنیم و خیلی خیزای خدید بسازیم.

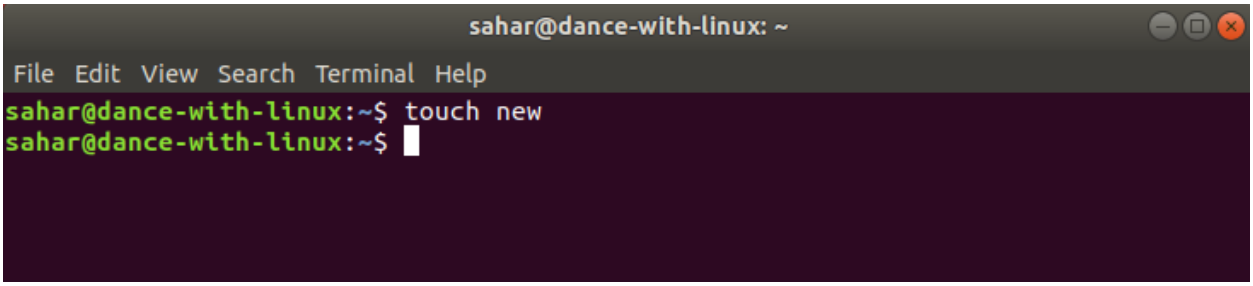
یکی از راه‌ها برای ساختن فایل این است که بگیم "هیچی" رو ریدایرکت کن به فایلی مثل khali.txt و اینجوری یک فایل خالی به اسم khali.txt ایجاد شده است.

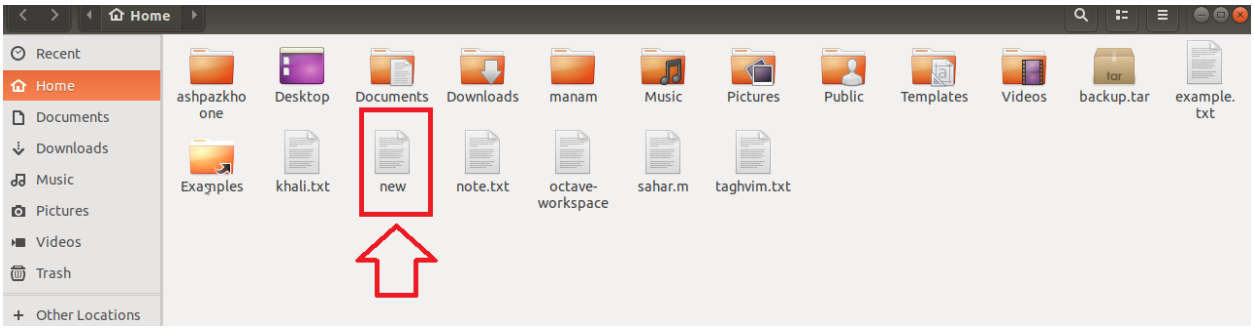


فایل ایجاد شد:



یک راه دیگر، استفاده از کامند touch است. مثلا اگر بزنم:





کار اصلی touch این است که آخرین زمانی که یک فایل modify شده را تغییر می‌دهد.

یعنی اگر برم دَر یخچال رو باز کنم، و کامند touch رو بزنم، زمان آخرین باری که در یخچال رو باز کردم، عوض می‌شه؟

آره. کامند تاچ یا همون touch ، به ازای هر فایلی که شما ذخیره می‌کنید، یک عالمه اطلاعات دیگر هم، مثل زمان ایجاد اون فایل، آخرین زمان تغییر اون فایل و... ایجاد می‌شود.

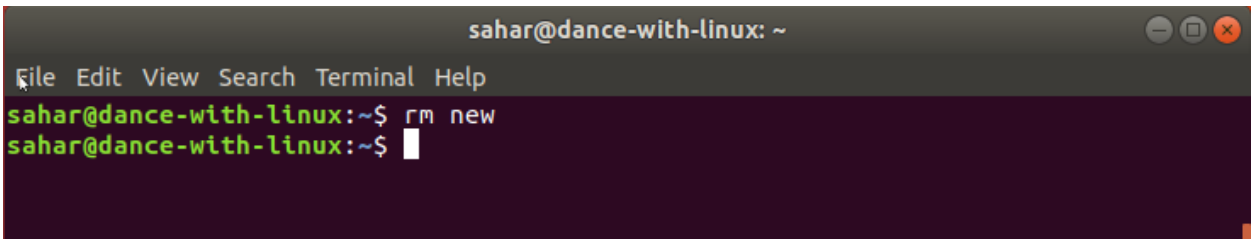
صاحب فایل رو هم نشون می‌ده. به اینا می‌گن meta data

ولی چرا همچین واکنشی نشان می‌دهد؟ یعنی چرا می‌توان برای ایجاد فایل از آن استفاده کرد؟ پاسخ ساده است. در رودروایسی گیر می‌کند! مثلا می‌گوییم meta dataهای فایل sahar را اصلاح کن، می‌گردد، فایل sahar را پیدا نمی‌کند. یعنی فایل sahar کلا وجود ندارد. پس آن‌را می‌سازد! نجابت یعنی همین.

کامند rm

بجز rmdir کامندی داریم به اسم rm

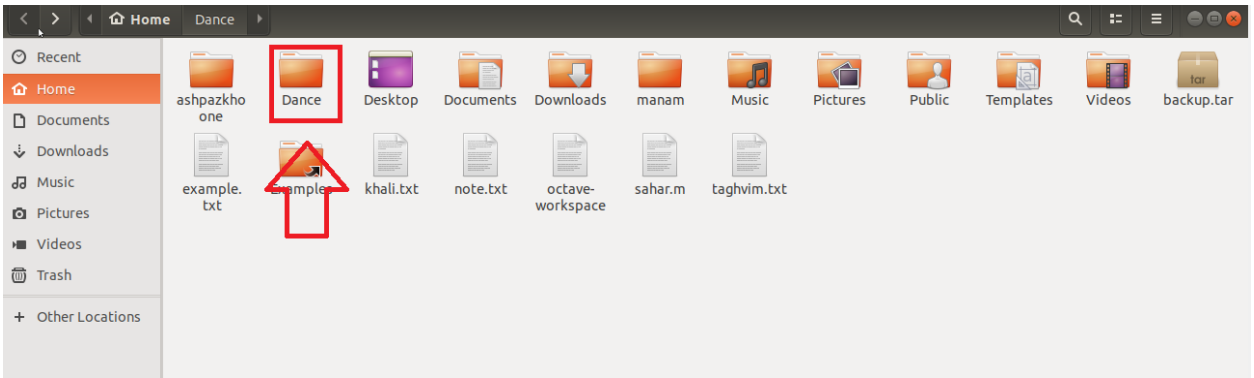
این کامند فایل پاک می‌کند اما درمقابل حذف یک دایرکتوری عاجز است. یعنی می‌توان نوشت:

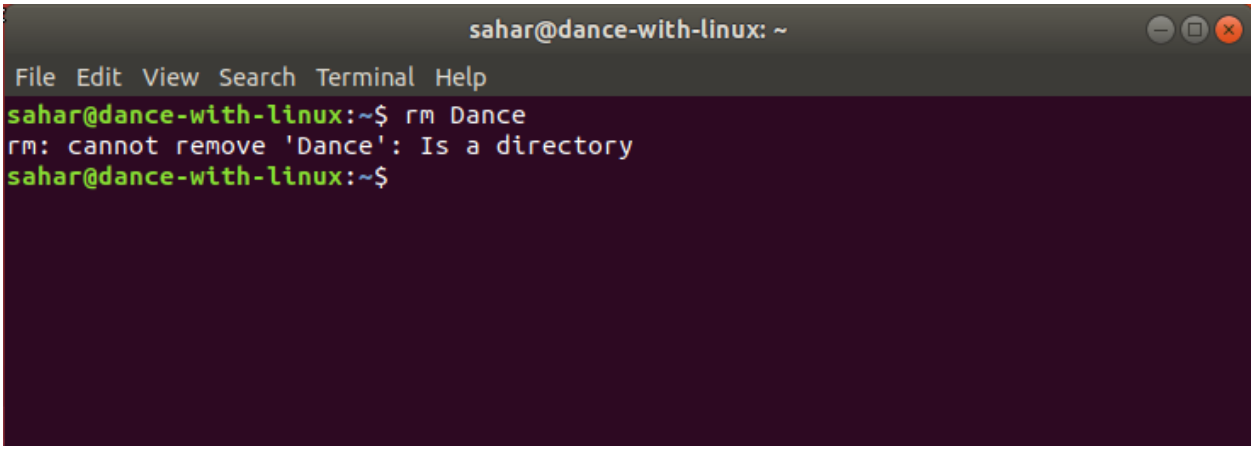


فایل را پاک می‌کند.

این کامند نمی‌تواند "دایرکتوری" پاک کند؛ منظور از اینکه دایرکتوری را پاک کنیم، یعنی تمام محتویات داخل آن پاک شود، و بعد هم خودِ فولدر پاک شود.

مثلا فرض کنید در دایرکتوری جاری، یک فولدر ساختیم به اسم Dance، اگر بخواهیم با دستور rm این دایرکتوری را پاک کنیم:

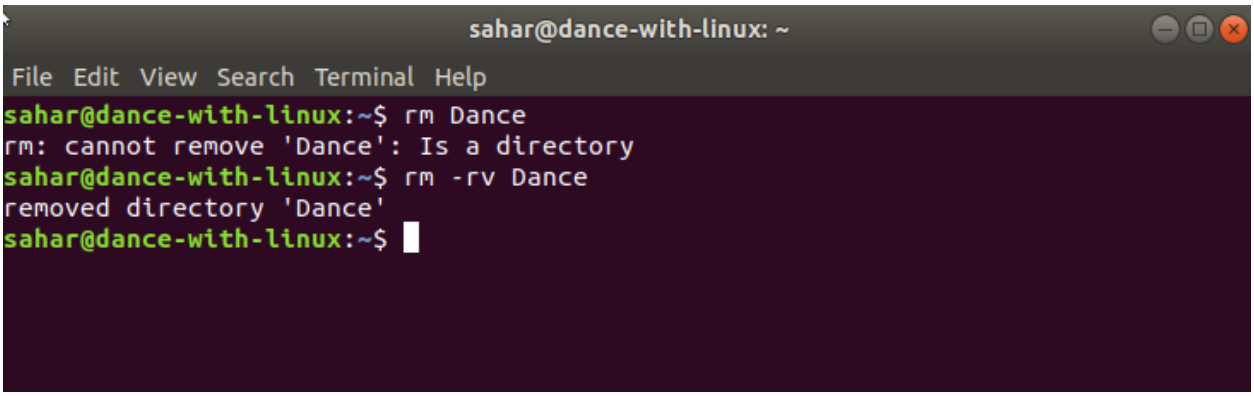




ارور می‌دهد که Dance یک دایرکتوری است و نمی‌تواند آن‌را پاک کند.

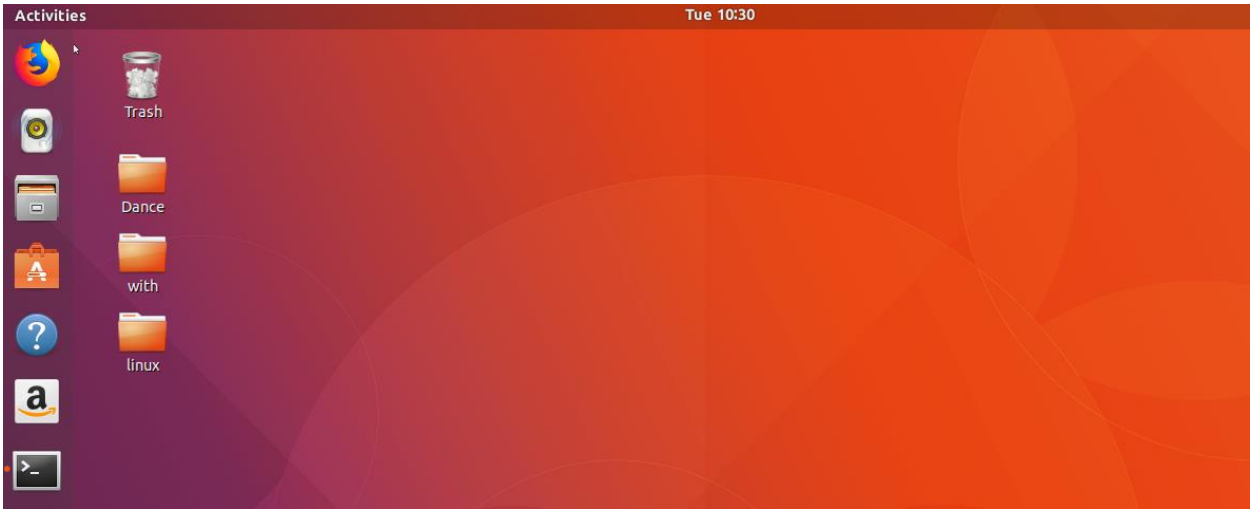
اما برای پاک‌کردن دایرکتوری Dance با استفاده از دستور rm می‌توان از آپشن این دستور، یعنی r- کمک گرفت.

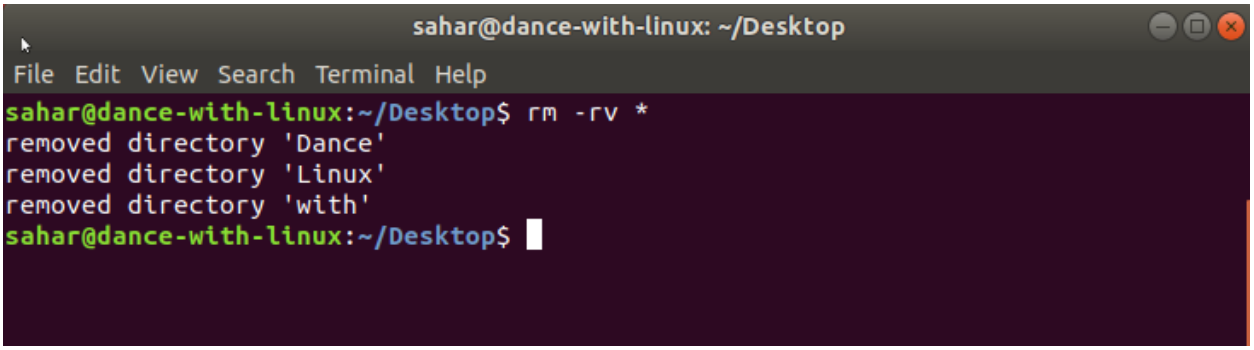
در واقع این آپشن در کنار کامند rm، اول محتویات داخل دایرکتوری Dance را پاک می‌کند و بعد هم دایرکتوری Dance را حذف می‌کند.



فان: rm هم بچه کم حرفیه برای به حرف درآوردنش از v- کمک گرفتیم، برای همین نوشتیم rv-

مثلا اگر سه تا فایل بسازم یکی به اسم dance و دیگری به اسم with و آخری به اسم linux و حالا بخواهم پاک‌کنم و یک پیغام هم نشان دهد می‌نویسم:





نکته: علامت * همه فایل‌های داخل دایرکتوری جاری (دایرکتوری‌ای که هستید) را پاک می‌کند.

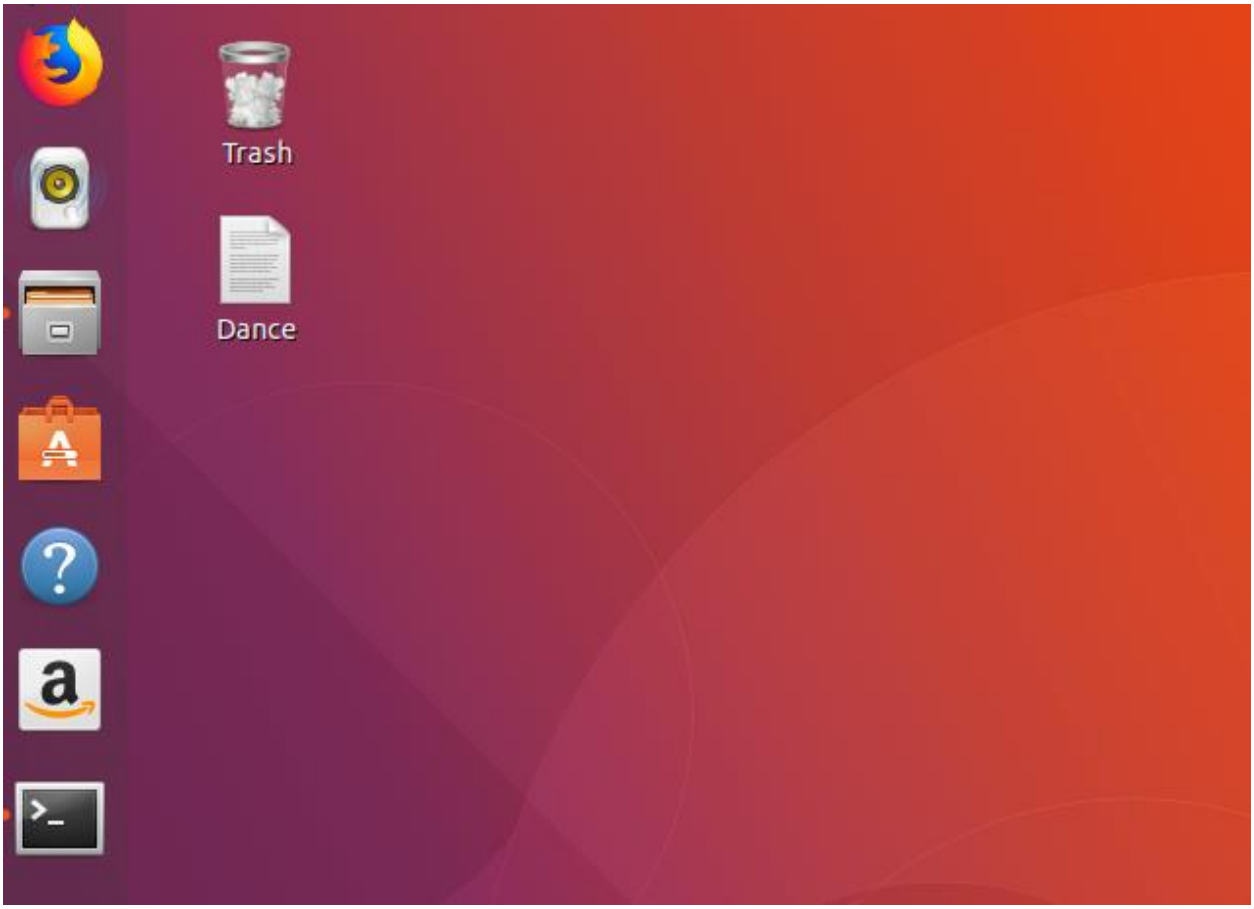
یه سوال بپرسم؟ تو همیشه قیافت اینجوریه، اگر خانواده ارث بپرسن پیا بریم مهمونی می‌ری؟ چجوری واکنش نشون می‌دی کلا؟

نه می‌گم شما برید من تیر خوردم.

کامند cp

cp کامندی است که برای کپی‌کردن فایل‌ها استفاده می‌شود. با آن می‌توان یک یا چندین فایل را در فولدری کپی کرد.

مثلا فرض کنید فایل Dance که بر روی دسکتاپ هست را می‌خواهیم در دایرکتوری home کپی کنیم:



```
sahar@dance-with-linux: ~
File  Edit  View  Search  Terminal  Help
sahar@dance-with-linux:~$ cp /home/sahar/Desktop/Dance /home/sahar
sahar@dance-with-linux:~$
```

یعنی بعد از cp دایرکتوری جایی که فایل در آن است را می‌دهیم بعد یک space یا همان فاصله خالی و بعد جایی که می‌خواهیم فایل در آن ذخیره شود.

آدرس اتاقی که هست رو بده و بگه می‌خوای توی کدوم اتاق کپی شه

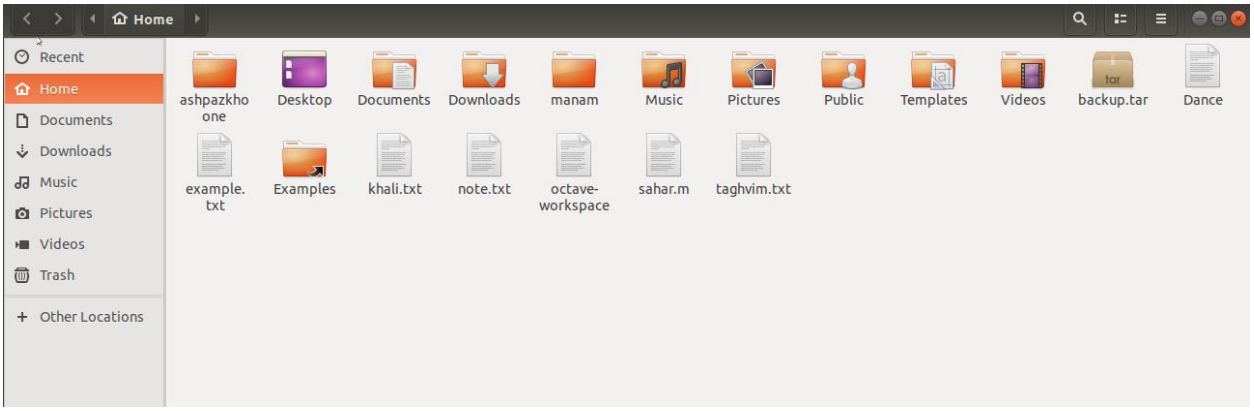
اگر من بودم از معشوقم ده‌تا کپی می گرفتم توی جاهای مختلف نگهش می‌داشتم

اشتباه می کنی خب خره، چیزی که بیوفته روی دورِ تکرار و همش ببینیش که دیگه دوسش نداری

خوشحالم که شماهم از این چیزا حالیتونه جناب!

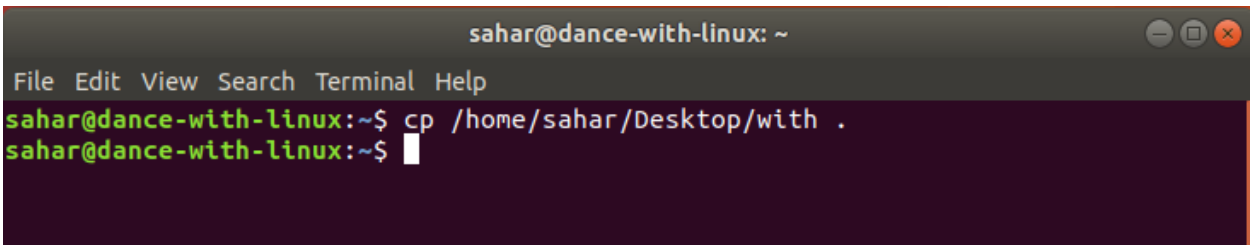
با خودم عهد بسته بودم که دیگه تو رو شیر فهم نکنم

عمت شیر فهم شه بی‌شخصیت... من خرفهم می‌شم افتاد؟؟؟

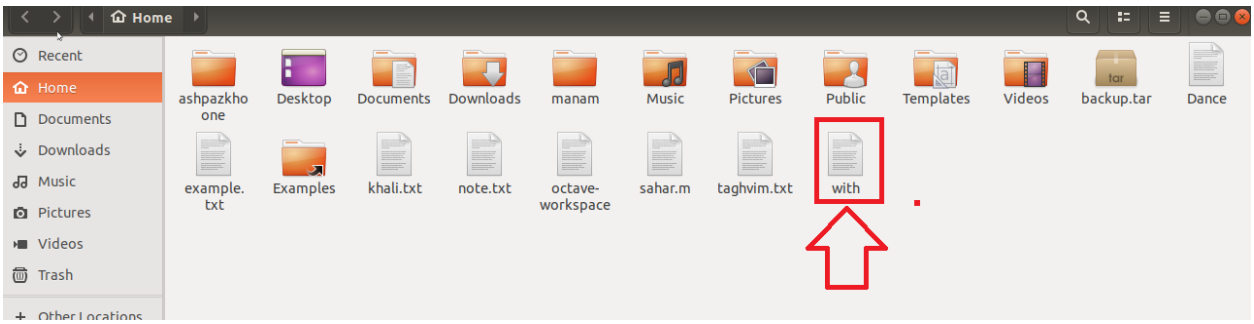


مثلا فرض کنید در دایرکتوری home هستیم، حالا می‌خواهیم از روی دسکتاپ، فایل with را داخل home که هستیم، کپی کنیم:

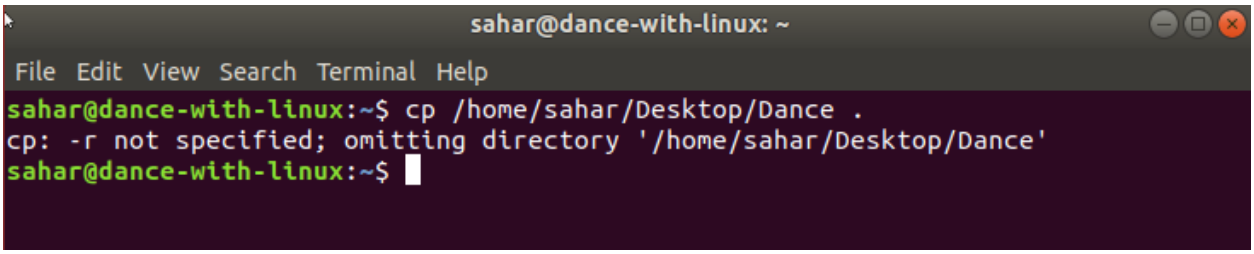
برای اینکار اول باید آدرس جایی که with در آن هست را بدهیم، بعد یک فاصله، حالا بجای اینکه آدرس home را که دایرکتوری جاری است بدهیم، بجایش یک نقطه می‌گذاریم:



یعنی برای دستور cp نیازی نیست که آدرس دایرکتوری جاری را بدهید، کافی است یک نقطه بجایش بگذارید.



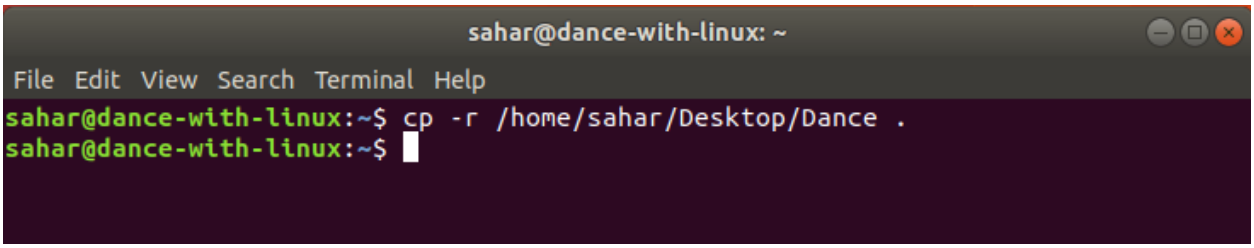
حالا مشکلی که با cp وجود دارد، این است که cp فایل جابجا می‌کند اما دایرکتوری نه.



ارور می‌دهد.

کپی‌کردن برای یک فایل معنا دارد. یعنی یک فایل داریم، که داخل این فایل محتوایی وجود دارد حالا می‌توانیم آن‌را جابجا کنیم. اما دایرکتوری خودش وجود ندارد، صرفا در tree یک اسم دارد و کپی‌کردن مفهومی برایش ندارد.

اما وقتی به طور عادی حرف از کپی زدن می‌شود، یعنی توقع داریم یک فایل جدید ساخته شود و محتوای آن فایلی که می‌خواهیم درون آن هم کپی شود.برای اینکار یک آپشن r- کنار کامند cp می‌آید:



اگر هم تایپ کنیم cp -rv ، آپشن v پیغام نشان می‌دهد که چه چیزهایی را در کجا تایپ کرده است.

نکته: اگر فایلی که داریم جابجا می‌کنیم، از قبل در مقصد وجود داشته باشد، آن فایل بازنویسی می‌شود، یعنی پاک می‌شود و بعد فایل جدید جایگزین آن می‌شود؛ به عبارتی دیگر فایل قبلی پاک می‌شود! برای اینکه این اتفاق نیافتد از آپشن b- استفاده می‌کنند. یعنی با استفاده از این آپشن، حتی اگر قبلا هم این فایل در مقصد وجود داشته باشد، اسم قبلی عوض می‌شود و فایل جدید کپی می‌شود. در واقع شما هر دو ورژن را خواهید داشت.

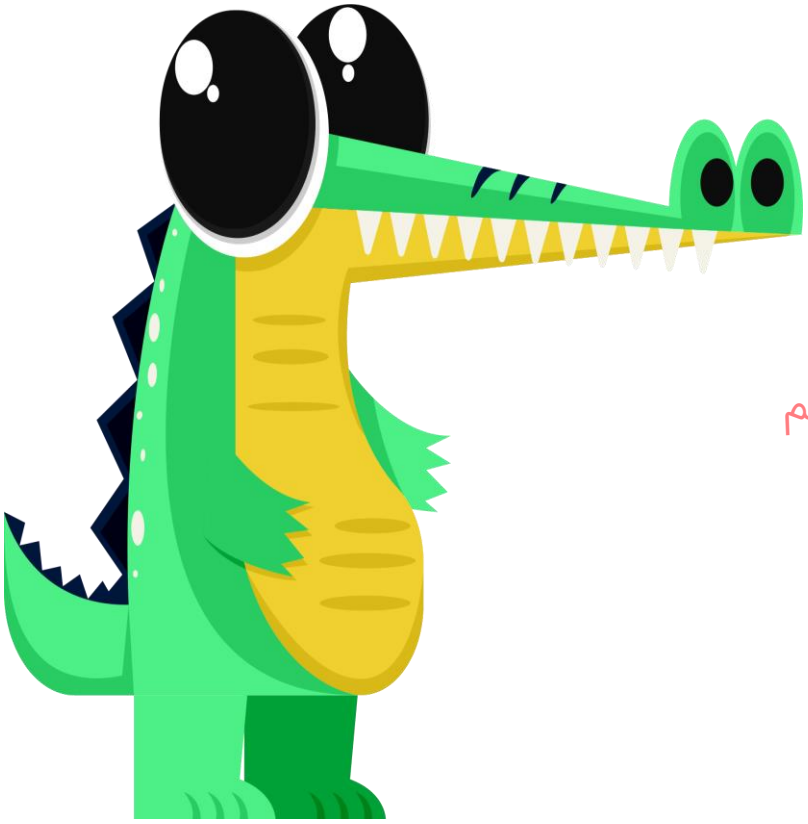
کرنل انقدر بحث پیچیده و عمیق‌تر و شیرینی داره که ما عملا توی این فصل چیزی در موردش نگفتیم

ولی کتاب در حال آپدیت شدنه، این‌ها رو پیشتر به چشم مقدمه و آشنایی خیلی خیلی کوچیک ببینین

کل کتاب هدفش آشنایی شما با لینوکسه

امیدوارم اگر وقت و عمری بود دوره یا کتابی روی سایت منتشر کنم که به طور اختصاصی و فنی (نه فان) روی توسعه‌ی کرنل لینوکس بحث کنه

فصل سوم: شل‌اسکریپت نویسی



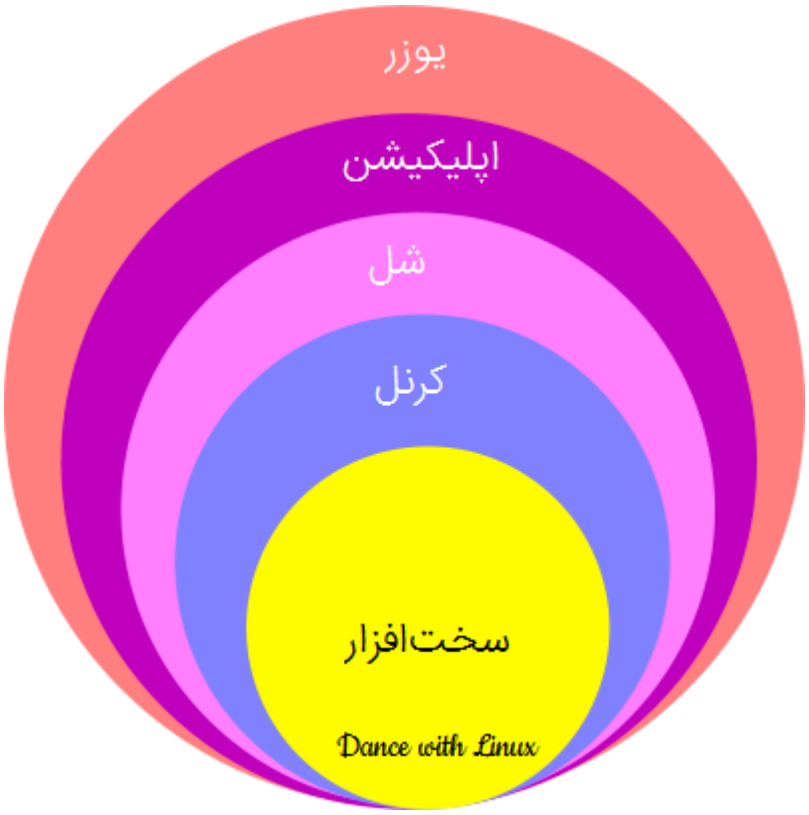
چجوری shell script بنویسم

یکم کاربردی‌تر

تا اینجای داستان، چیا فهمیدیم؟

هیچی، اینکه "خر" می تونه عاشق هر چیزی بشه

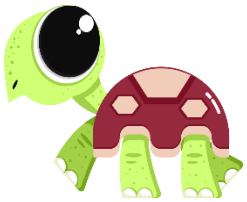
اینکه کرنل یه Interfaceایه بین hardware و application



با این رنگ‌ها اصلا معلوم نیست این کتابو یه دختر نوشته

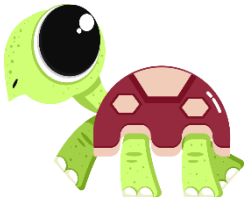
سخت‌افزار چیه؟

خیلی ساده هر چیزی که بشه لمسش کرد.



بعد از سخت‌افزار، کرنله. کرنل در واقع یه برنامه است که داخل سیستم عامل شما ذخیره شده و دستورات یا همون commandها رو از shell می گیره

حالا shell چیه؟



Shell یجور GUI می‌مونه که ما توی ویندوز یا توی لینوکس داریم. یا اگر لینوکسی داری که از GUI استفاده نمی‌کنه، یه ترمینال داری که روش bash shell یا C shell رو داره که ما توش یکسری command اجرا می‌کنیم، این کامندارو ور می‌داره می‌بره می‌ده به کرنل تا کرنل با سخت‌افزار حرف بزنه و بگه شما چی می‌خواین.

بعد از اون، اپلیکیشن‌ها رو داریم. مثلا یه مرورگر، تقویم و ادویه به مقدار لازم

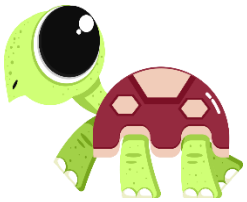
ته همه‌ی این‌ها هم کاربر نشسته بر اون تخت سلطنت

پس shell به همراه Kernel یک packageای ایجاد می‌کنند که به آن Operating System یا سیستم عامل می‌گویند. این سیستم‌عامل هر چیزی می‌تواند باشد؛ لینوکس، ویندوز، یونیکس، مک و...

هر سیستم‌عاملی این کرنل را دارد تا با سخت‌افزار صحبت کند، و یک shellای دارد که به زبان C نوشته شده.

Shell+ Applicationها می‌شه نرم‌افزار یا همان Software

پس در واقع کرنل همین مواردی بود که به توضیح آن پرداختیم. با هر سیستم‌عاملی که کار کنیم، به یک کرنل نیاز داریم تا با سخت‌افزار بتواند صحبت کند.



Shell شبیه یک ظرفی می‌مونه که شما اطلاعاتتون رو در اون قرار می‌دید. Shell یه Interface بین یوزر با کرنله یا سیستم‌عامله. مثلا ویندوز یه سیستم‌عامله. وقتی با GUI کار می‌کنین، با دابل‌کلیک کردن یه برنامه باز می‌کنین یا اجرا می‌کنین یعنی دارین یه کامند رو اجرا می‌کنین. حالا می‌دونیم که

برای اینکه متوجه شویم که از چه shellای استفاده می‌کنیم می‌توانیم از echo $0 استفاده کنیم.

یا با اجرا کردن کامند cat /etc/shells تمام shellهایی که روی سیستم‌عامل شما نصب شده را نشان می‌دهد.

راه دیگر برای پیداکردن shellها استفاده از دستور cat /etc/password

این دستور نشان می‌دهد که یوزر شما از چه shellای استفاده کرده است.

```
sahar@dance-with-linux: ~
File  Edit  View  Search  Terminal  Help
sahar@dance-with-linux:~$ echo $0
bash
sahar@dance-with-linux:~$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
sahar@dance-with-linux:~$
```

shellها

Shell گرافیکی و shell کامندلاینی

اگر سیستم‌عامل شما لینوکس باشد و بخواهید از یک محیط گرافیکی یا GUI استفاده کنید، GUIهای در دسترس، یکی گنوم یا Gnome است و دیگری KDE. هر دوی این‌ها محیط‌های GUIای برای دسکتاپ به شمار می‌روند و هر دوی این‌ها هم shell هستند.

در واقع وقتی به این محیط‌های گرافیکی لاگین می‌کنید، (وقتی وارد این محیط‌های گرافیکی می‌شوید) در یک shell هستید و با آن دستورات خودتان را اجرا می‌کنید.

چجوری؟

با موس و کیبورد دیگه، مثلا کلیک می‌کنید، drag & drop می‌کنید.

بجز shell گرافیکی یک shell کامندلاینی هم وجود دارد. وقتی که در سیستم‌عامل با GUI طرف نباشیم یا آن را نداشته باشیم با shell کامند لاینی سر و کار داریم.

قبلا هم توضیح داده شد که shellهای کامندلاینی مختلفی وجود دارد اما به طور دیفالت، shell **اکثر** لینوکس‌ها bash است.

Shell script چیه؟

خیلی ساده، فرض کن یه کاسه داری که توش دستورالعمل‌هایی که می‌خوای رو میریزی. مثلا میگی این کارو بکن، اون کارو نکن. یا اگر این کارو کردی بعدش این کارو کن. اگر اونیکی جواب نداد فلان کارو انجام بده و... تهش هم به shell می‌گی همه‌ی این دستورالعمل‌ها رو انجام بده عمو ببینه. Shellهم به ترتیب اون‌هارو اجرا می‌کنه.

فکر کنم ننه منم منو با shell script اشتباه گرفته بوده

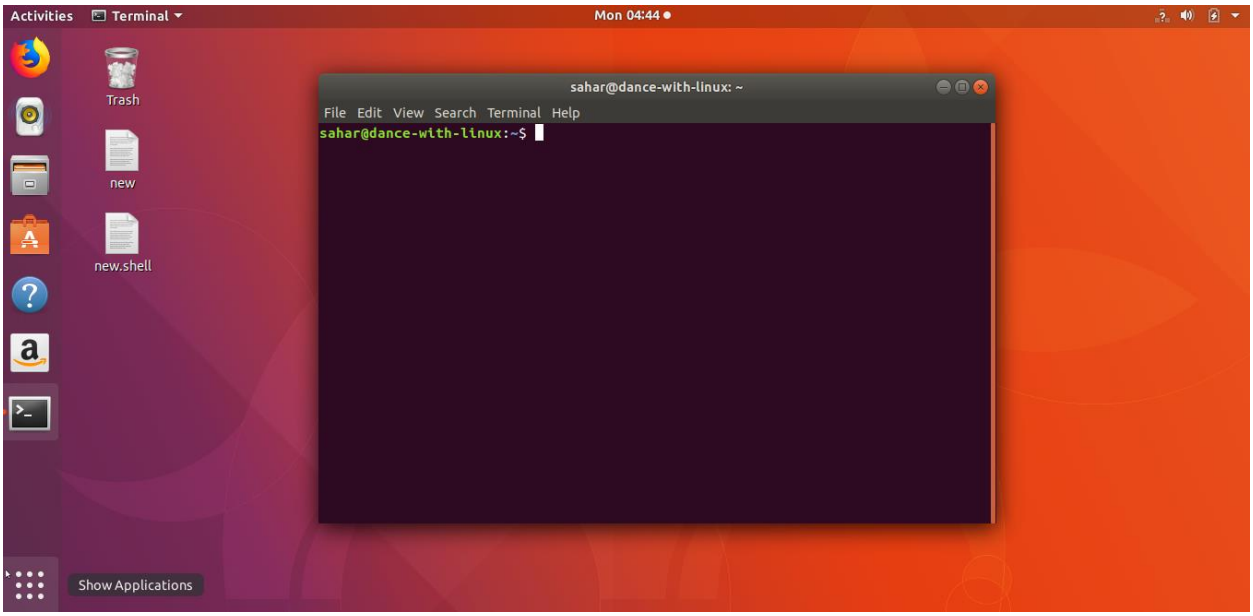
به عنوان مثال وقتی در ویندوز، یک فایلی را می‌خواهید کپی کنید، و جای دیگر paste کنید، راست کلیک می‌کنید و کپی می‌کنید یا اینکه با ترکیب ctrl+c و بعد ctrl+v فایل را

کپی می‌کنید؛ این کار خودش یک فانکشن در shell به شمار می‌رود. اینکه چگونه این کپی‌کردن اتفاق می‌افتد همگی زیر سر فانکشنی است که در shell عهده‌دار انجام این دستور است.

این عمل کپی پیست‌کردن، یک shell script به شمار می‌رود. چه کسی آن‌را نوشته؟ یک دولوپر.

این به این معناست که شما هم می‌توانید shell scriptهای خودتان را بنویسید و اجرا کنید، ما هم در این فصل همین قصد را داریم.

قصد داریم یکسری یکسری shell scriptهای واقعی بنویسم و اجرا کنیم.



در تصویر چند shell می‌بینید؟

هیچی

سه تا

چرا سه تا؟

پس چند تا؟

دو تا، یکی محیط گرافیکی Gnome یکی هم shell کامند

نوچ، تو مو می‌بینی و من پیچشش مو

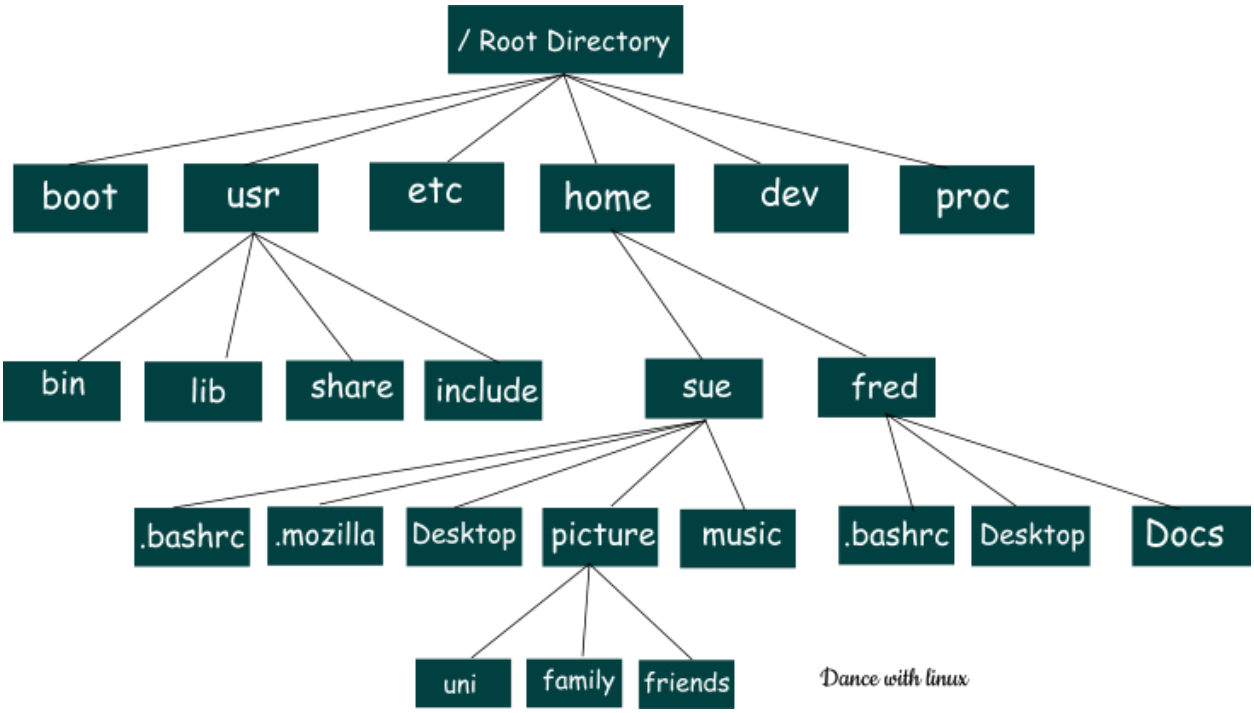
چجوری یک **shell script** را اجرا می‌کنند؟

وقتی یک shell script نوشتیم باید مسیر فایلی که shell script در آن هست را به shell بدهیم. یعنی ما این shell script را در یک فایلی نوشتیم و این فایل را یک‌جایی گذاشتیم. برای اجرا کردن آن باید مسیر جایی که فایل در آن است را به shell معرفی کنیم.

اگر برگردیم سر ویندوز، در ویندوز درایو داریم مثل درایو C, D, E, F برای آدرس‌دهی هم سراغ آن‌ها می‌رویم.

اما در لینوکس چنین خبری نیست. سیستم‌عامل‌های مبتنی بر یونیکس، یک فایل سیستم بیشتر ندارند که همان دایرکتوری root است و بقیه مثل شاخه به این دایرکتوری root وصل می‌شوند.

یکجور دیگر بیان کنم: در لینوکس فایل‌ها یا همان دایرکتوری‌ها در یک قالب درختی قرار دارند. این درخت یک شاخه‌ی اصلی دارد که شاخه‌های دیگر به این شاخه‌ی اصلی متصل‌اند. این شاخه‌ی اصلی در لینوکس، دایرکتوری root است. این دایرکتوری روت، یکسری زیرشاخه‌ها و دایرکتوری‌هایی دارد که باز هم همان زیرشاخه‌ها، زیرشاخه‌های دیگر دارند.



مسیرها

دو جور مسیر (path) داریم:

- Absolute Path
- Relative Path

Absolute Path

در مسیریابی مطلق یا همان Absolute Path پله‌پله پایین می‌آییم تا به دایرکتوری‌ای که می‌خواهیم برسیم، برسیم. فرض کنید بالای پله‌هاییم و فایل مورد نظر ما در یکی از پله‌های پایینی است (زیرشاخه‌های زیری دایرکتوری root). برای برداشتن آن فایل، پله‌ها را یکی یکی آدرس می‌دهیم و پایین می‌آییم.

مسیرهای Absolute با / شروع می‌شوند. / نماد دایرکتوری root است.

عین کوه‌نوردی می‌مونه، توی هرجای مسیر کوه‌نوردی که باشی، بازم می‌ره از قله یعنی دایرکتوری root شروع می‌کنه و دونه‌دونه دایرکتوری‌هارو میاد تا به اونی که میخوای برسه.

دونه‌دونه دونه‌دونه یه دایرکتوری رو پشت بومه، یه جوری میزون می‌کنه که ما رو به هم برسونه. دل گرفتاره، عاشق یاره، من نه مستم و هوشیار، یه حال جدیدی بینمونه. یه چیزایی توی چشماته که دلم می‌لرزونه

این که چشاش چپه، کجاتو دقیقا می‌لرزونه؟

مثلا اگر بخواهیم با روش Absolute Path به دایرکتوری /usr/bin برویم، تایپ می‌کنیم

```
cd /usr/bin
```

(در فصل قبل یاد گرفتیم که برای تغییر یک دایرکتوری از کامند cd استفاده می‌شود.)

در واقع با Absolute Path یا مسیردهی کامل، ما آدرس کامل دایرکتوری که می‌خواهیم را می‌دهیم.

چهارراه دلتنگی/خیابان تنهایی/ کوچه دوستی/ بن‌بست عشق/ پلاک محبت/ ساعت دلتنگی/ یک‌ربع مانده به دلهره

Relative Path

ولی Relative Path **برنمی‌گردد** از فایل root شروع کند. از دایرکتوری که در آن حضور داریم شروع می‌کند تا به دایرکتوری مد نظر برسد و با علامت / شروع **نمی‌شود**.

یعنی در Relative Path یا مسیردهی نسبی، ما **نسبت** به دایرکتوری‌ای که هستیم، آدرس می‌دهیم.

مثلا با Relative Path اگر بخواهیم به /usr/bin برویم:

زی ../../عسق/ذهد

تو هم سرایت کرد؛ کیبوردتو انگلیسی کن.

شبیه اسم پروفایل می‌مونه

اسم پروفایل؟

آره دیگه اینو روزا می‌ری پروفایلارو می‌بینی همش ای‌دی‌ها پسری رمزه. پیشتر هم رمزهای ناسا

cd ../../usr/bin

```
sahar@dance-with-linux: /usr/bin
File  Edit  View  Search  Terminal  Help
sahar@dance-with-linux:~$ cd ../../usr/bin
sahar@dance-with-linux:/usr/bin$
```

حالا چرا دوبار از /.. استفاده کردی؟

چون دایرکتوری‌ای که الان هستیم، دو سطح از دایرکتوری root فاصله دارد.

../../../../../../../

چته؟

می‌خوام بگم خیلی دوریم

خلاصه باید به shell بگید که script شما کجاست.

قول‌وقرارها سر اسم‌گذاری اسکریپت‌ها

1.اولین موردی که بهتره به آن توجه داشت، ایجاد دایرکتوری برای اسکریپت‌هایی است که می‌نویسیم.

بهش توی این خونه بزرگ (لینوکس) یه اتاق (یه فولدر یه دایرکتوری) بده، نذار آواره باشه اینور و اونور. اگر متعلق به یه گروه خاصیه (یه پروژه) توی همون جا بهش جا بده. همانا از آواره کنندگان نباشید.

یک اسم برای آن دایرکتوری انتخاب کنید، مثلا Scripts یا myScripts

```
sahar@dance-with-linux: ~/Desktop
File  Edit  View  Search  Terminal  Help
sahar@dance-with-linux:~/Desktop$ ls -ltr
total 8
drwxrwxr-x 2 sahar sahar 4096 Oct 23 10:56 Dance
drwxrwxr-x 2 sahar sahar 4096 Jan 17 05:22 script
sahar@dance-with-linux:~/Desktop$
```

2. برای این بچه (اسکریپت) اسم درست و حسابی انتخاب کنید.

یه اسمی که نشون بده کارش چیه، نذارید سوپری ممدرضا. واضح بنویسید سوپری برادران محمدرضا بجز برادر وسطیه. اون درس نخوند پولدار شد بنز سوار شد. ولی من درس خوندم بعد با پولش یه وانت خریدم و گوجه فروختم.

خلاصه یجوری کدنویسید که صبح بنویسید بعد عصر نفهمید چی نوشتید.

اسمی برای اسکریپت انتخاب کنید تا معرف کاری باشد که انجام می‌دهد.

3. اسکریپت با shell. تمام شود

این مورد از لحاظ مهم است که ممکن است شما اسکریپت برای shellهای مختلف بنویسید. مثلا اسکریپت شما برای Korn Shell یا C shell یا Bash باشد.

اگر اسکریپتی با اسم helloWorld در bash shell نوشته شده است، برای اجرای آن باید نوشت:

helloWorld.bash

این مورد بیشتر اختیاری است، اگر رعایتش کنید برای اجرا یا ران‌کردن اسکریپت لازم نیست حتما آدرس‌دهی کنید، اگر در محیط گرافیکی باشید با دوبار کلیک‌کردن روی اسکریپت برنامه اجرا خواهد شد.

Script File Permision

در لینوکس برای فایل‌ها، اجازه‌ی دسترسی صادر می‌گردد. یعنی هر فایل مطابق با یوزری که صاحب آن فایل است، اجازه‌ی دسترسی به فایل یا shell scriptها و اجرا کردن آن‌ها را دارد.

یعنی صاب فایل باس باشه تا بهتون اجازه بده یا اجازه رو بگیرین

هر کدام از یوزرها نیز، به یک گروه کاربری تعلق دارد. هر کدام از این گروه‌ها به یکسری مجوزها دسترسی دارند که برای کل یک گروه، این اجازه‌ها یکسان است.

پس پرمیشن‌ها بسته به اینکه اون کاربر توی کدوم گروه باشه، بحث می‌شه

چند نوع پرمیشن داریم؟

سه نوع پرمیشن در لینوکس تعریف می‌شود:

read.1

write.2

execute.3

این سه پرمیشن غالبا با حروف اولشان شناخته می‌شوند. یعنی read با r، write با w و execute با x .

execute کجاش با x شروع می‌شه؟

واسه همینه می گم تو مو می‌بینی و من پیچشش مو

ولی کلا در لینوکس برای مدیریت راحت‌تر پرمیشن‌ها، این سه مورد را با عدد مشخص می‌کنند. یعنی عدد 4 برای پرمیشن read، عدد 2 برای پرمیشن write و عدد 1 برای پرمیشن execute

read	write	execute
r	w	**e**
4	2	**1**

با این جدول، متوجه می‌شویم که عدد 7 یعنی آن گروه یا یوزر پرمیشن read, write, execute را دارد. عدد 5 نشان می‌دهد که آن گروه یا کاربر فقط پرمیشن read, execute را دارد. عدد 6 نشان می‌دهد که آن گروه یا کاربر فقط پرمیشن read, write را دارد.

آهاااان پس برای همینه که عدد 3 رو نذاشتن؛ چون اگر 3 هم توی جدول بود اوضاع یه هم می‌ریخت، چون جمع یک و دو می‌شه سه و این یعنی پرمیشن write و execute با همدیگه میتونستن یه پرمیشن دیگه بسازن!

این پرمیشن‌ها سه گروه را زیر سلطه دارند:

1. user که با حرف u شناخته می‌شود
2. group که با g شناخته می‌شود
3. others که با o شناخته می‌شود

آخه سومیه چی می‌گه؟ یا آدم یوزره یا گروپ دیگه

هر چی توی یوزر و گروپ نباشه میشه آدرز.

پس تا اینجای کار، سه تا گروه داریم، سه تا مجوز. هر گروه بسته به مجوزهایی که براش تعریف می‌شه اجازه‌ی دسترسی به فایل‌ها و دستکاریشون داره

دستورات برای پرمیشن‌ها

حالا سوالی که پیش می‌آید این است که دستورات یا commandها برای تغییر پرمیشن‌های ایجاد شده برای یک گروه یا یک یوزر چیست؟

یعنی چجوری می‌شه یه پرمیشن رو دستکاری کرد؟

سه command برای این کار وجود دارد، یک کامند برای تغییر پرمیشن گروه، یک کامند برای تغییر پرمیشن یوزر، یک کامند برای تغییر پرمیشن‌هایی که روی یک فایل وجود دارد:

chown

از این دستور برای تغییر مالکیت یک **کاربر**، یک فایل یا یک دایرکتوری استفاده می‌شود.

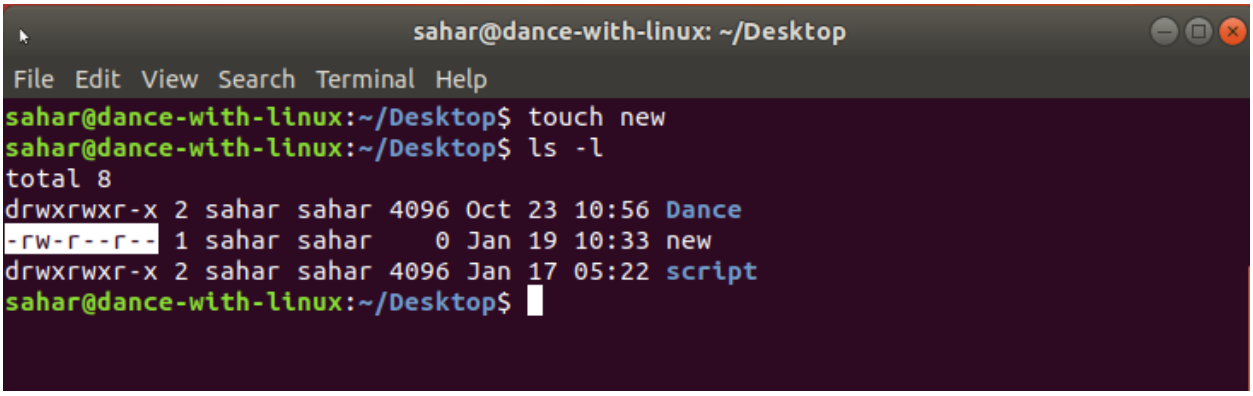
chgrp

از این دستور برای تغییر مالکیت یک **گروه**، یک فایل یا یک دایرکتوری استفاده می‌شود.

chmod

از این دستور برای عوض‌کردن پرمیشن‌هایی است که از قبل روی یک فایل وجود داشته است.

در تصویر زیر با دستور touch یک فایل در دسکتاپ ساخته شده (فایل new)، برای اینکه پرمیشن‌های این فایل را چک کنیم از دستور ls -l استفاده می‌کنیم:



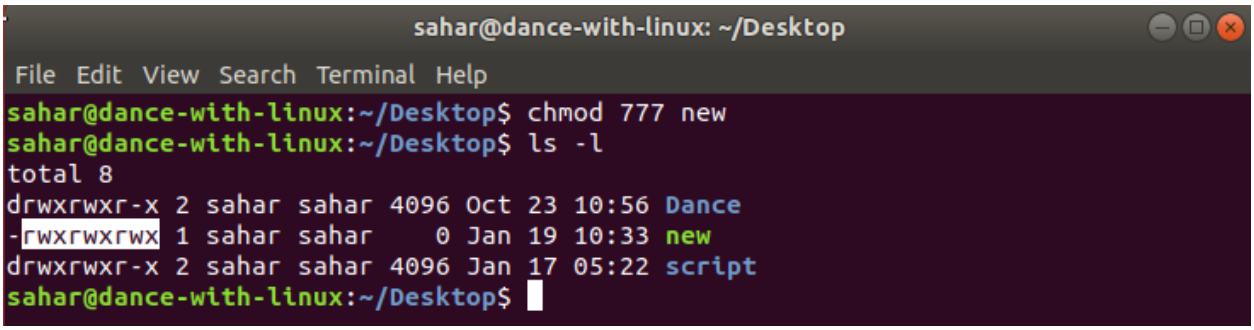
هایلایت سفیدرنگ، پرمیشن‌های فایل new را نشان می‌دهد. یعنی می‌گوید:

1.کاربری که این فایل را ساخته اجازه‌ی write و read را دارد.

2.گروهی که این کاربر به آن متعلق است تنها اجازه‌ی read دارد (دومی از سمت چپ).

3.others (دیگران) تنها اجازه‌ی read دارند.

اگر بخواهیم هر سه پرمیشن read و write و execute را به user و group و others بدهیم، از دستور chmod استفاده می‌کنیم:

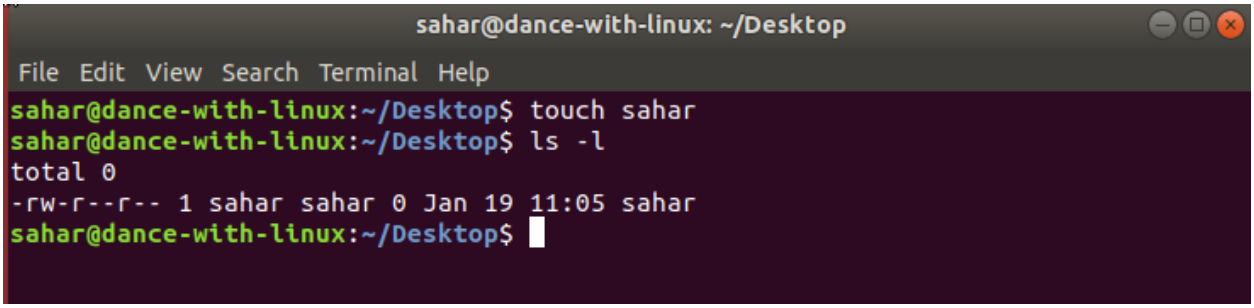


با دستور chmod 777 هر سه پرمیشن را به فایل new داده شد، هایلات سفیدرنگ همین را نشان می‌دهد.

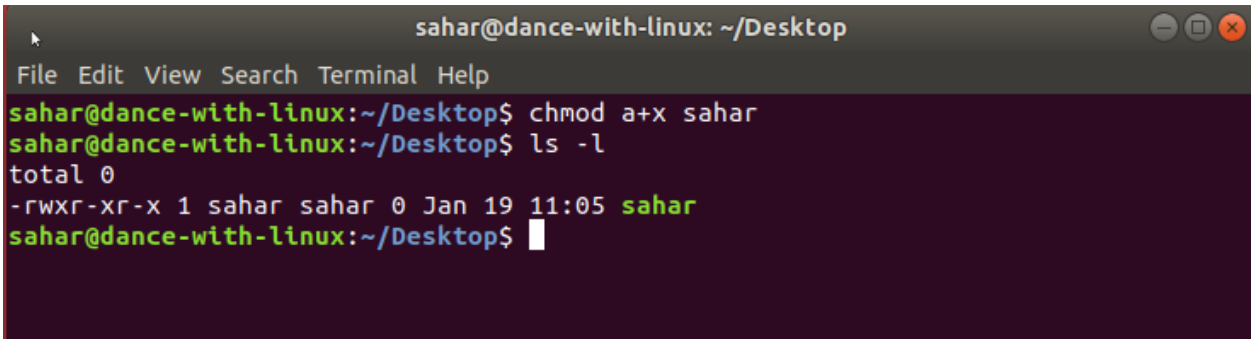
به طور دیفالت وقتی یک فایل یا یک اسکریپتی را می‌سازید، پرمیشن execute به آن داده نمی‌شود دلیل آن هم این است که لینوکس نمی‌خواهد شما برای یک لحظه دچار اشتباه شوید و به اشتباه چیزی را execute کنید. برای همین خود شما باید این پرمیشن را بدهید.

راه دیگری هم برای دادن پرمیشن execute یا x به هر سه یوزر و گروه و others وجود دارد؛ و آن استفاده از دستور chmod a+x script-name است. (a برای all)

مثلا در همان دایرکتوری دسکتاپ، فایلی می‌سازیم به اسم sahar:



به طور دیفالت می‌بینیم که برای فایل sahar پرمیشن x داده نشده. برای دادن این پرمیشن بجز راهی که قبلا گفتیم، می‌توان نوشت:



در اینجا می‌بینیم که پرمیشن x به هر سه داده شده است.

در مورد ساختن گروه‌ها

این قسمت حکم پیشتر بدانید داره

فایلی که یوزرها در آن قرار دارند، فایل /etc/passwd می‌باشد. در این فایل می‌توان یوزر جدید تعریف کرد و یا یوزرها را تغییر داد.

/etc/passwd/group نیز، گروه‌ها را نشان می‌دهد.

یک یوزر می‌تواند عضو چندین گروه باشد، اما نکته‌ی مهمی که وجود دارد این است که هر یوزر باید به صورت primary (اصلی) عضو یک گروه باشد. یعنی اگر بخواهیم یک یوزر اضافه کنیم، از قبل باید یک گروهی وجود داشته باشد که این یوزر را داخل آن گروه قرار دهیم.

اگر موقع تعریف یک یوزر، مشخص نکنیم که این یوزر برای چه گروهی است، به صورت دیفالت لینوکس یک گروه می‌سازد و این یوزر را داخل آن می‌گذارد.

بیایید در عمل ببنیم، برای این کار ابتدا یک shell روت باید بگیریم در غیر این صورت اجازه‌ی دسترسی را نخواهد داد، برای گرفتن شلِ روت، تایپ کنید:

```
sudo –s
```

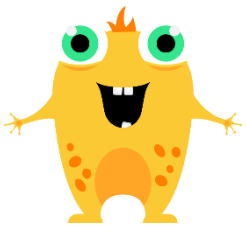
با زدن پسورد، شلِ روت آماده است، برای دیدن گروه‌ها تایپ کنید:

```
sudo cat /etc/group
```

با این لیست گروه‌های تعریف‌شده را خواهیم دید:

```
root@dance-with-linux: ~
File  Edit  View  Search  Terminal  Help
root@dance-with-linux:~# cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,sahar
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:sahar
floppy:x:25:
tape:x:26:
sudo:x:27:sahar
audio:x:29:pulse
dip:x:30:sahar
```

هر خط در این فایل نشان‌دهنده‌ی یک گروه است. هر گروه یک اسم دارد که اول خط نوشته شده، مثل man یا proxy یا news، برای هر گروه یک پسورد وجود دارد و هر گروه یک id دارد که به آن GID گفته می‌شود.

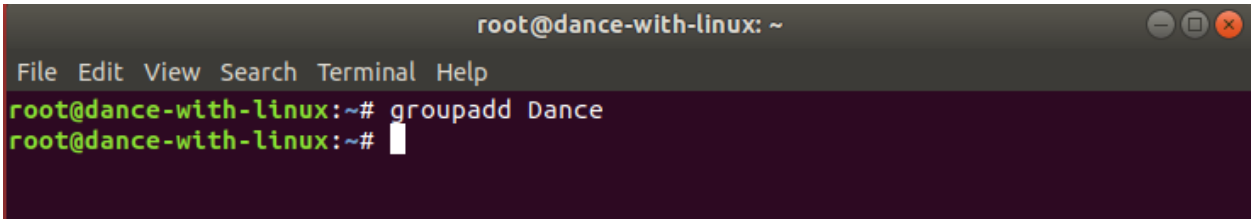


ما گروه‌ها رو با اسمشون می‌شناسیم و حرف می‌زنیم باهم یعنی مثلا می‌گیم گروه news یا گروه man؛ ولی سیستم با اون GID گروهه رو می‌فهمه. به تصویر نگاه کن. گروه news، جی‌ای‌دیش عدد 9. اطلاعات یه گروه با : از هم جدا می‌شه.

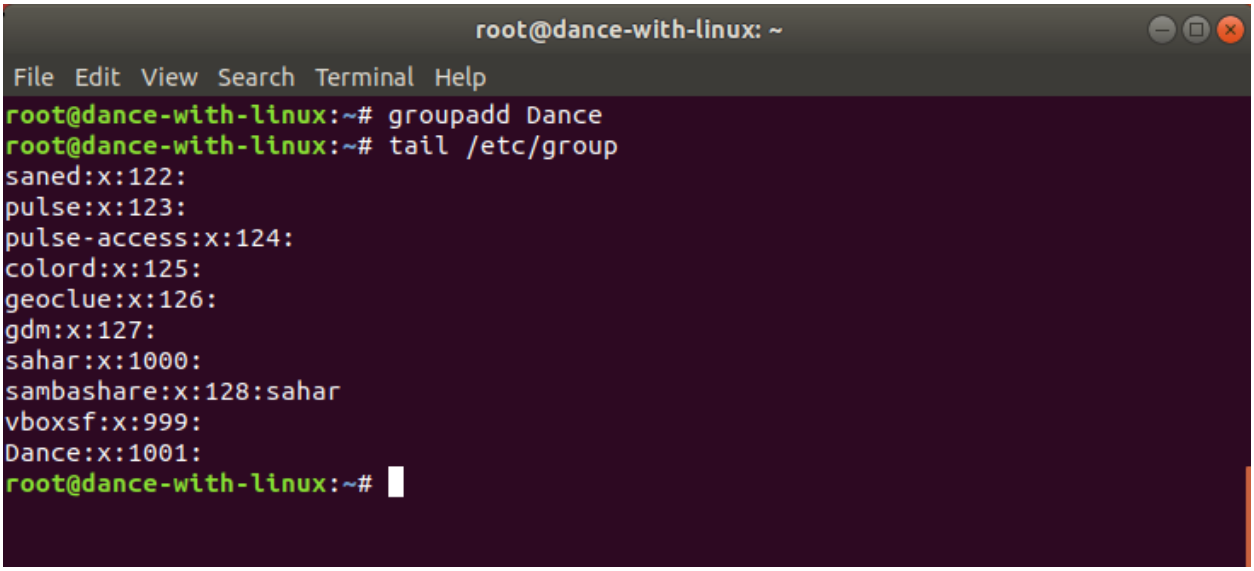
group name: password : gid: ...

اون سه نقطه آخر می‌گه یکسری داده‌های دیگه هم وجود داره، مثلا id یوزرهایی که عضو این گروهن و...

با کامند groupadd یک گروه جدید را می‌توان ساخت. مثلا گروه Dance را اینگونه بسازیم:



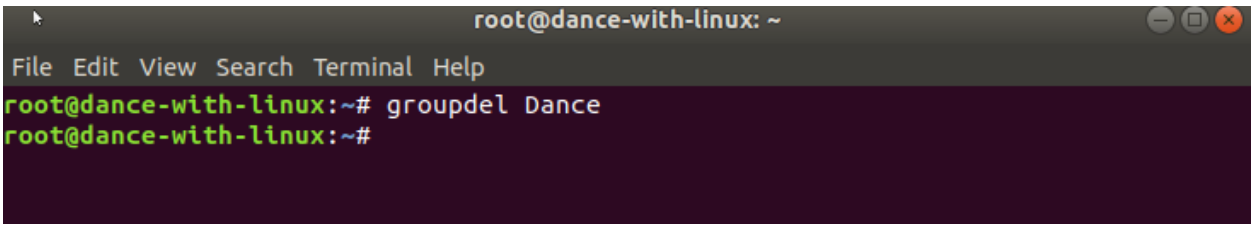
برای دیدنش:



می‌بینیم که در خط آخر، گروه Dance با جی‌آی‌دی 1001 ساخته شده است و هیچکس هنوز عضو آن نیست.

حذف گروه

با کامند groupdel گروه را می‌توان پاک کرد:



فقط یک نکته هست، یوزرهایی که در این گروه عضو بودند و حالا گروه حذف شده، عضویت آن‌ها باطل می‌شود.

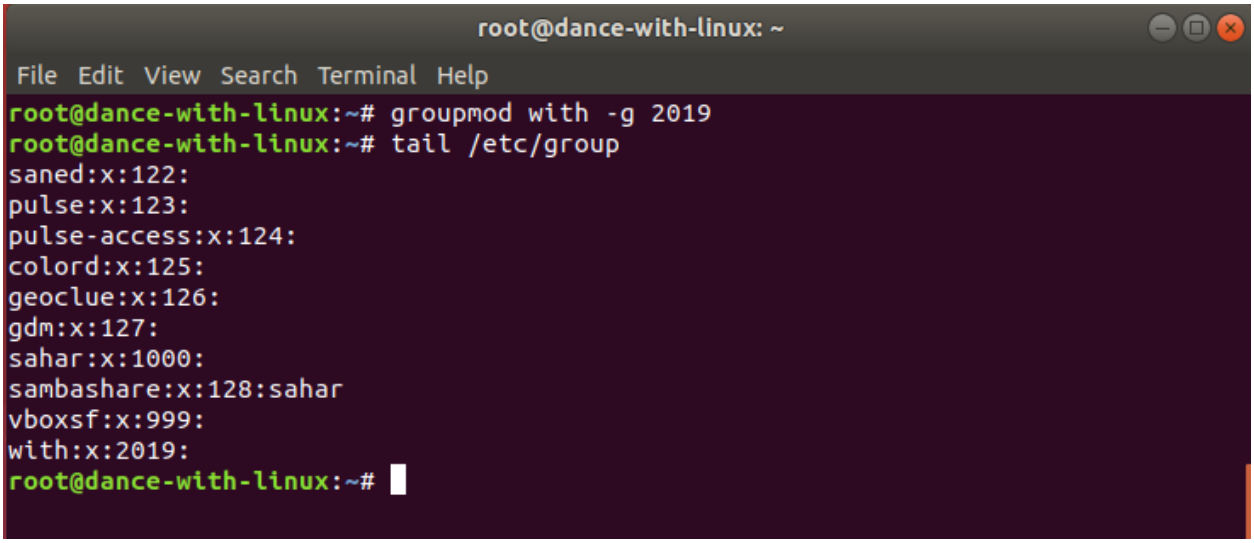
اما اگر یوزری به صورت primary عضو این گروه بود، موقع حذف گروه error خواهیم داشت. در این صورت برای آن کاربر باید یک گروه دیگری را به عنوان primary انتخاب کنید و سپس گروه را حذف کنید.

از دستور groupmod می‌توان برای تغییر گروه مثلا تغییر اسم گروه استفاده کرد:

```
root@dance-with-linux: ~
File  Edit  View  Search  Terminal  Help
root@dance-with-linux:~# groupmod Dance -n with
root@dance-with-linux:~# tail /etc/group
saned:x:122:
pulse:x:123:
pulse-access:x:124:
colord:x:125:
geoclue:x:126:
gdm:x:127:
sahar:x:1000:
sambashare:x:128:sahar
vboxsf:x:999:
with:x:1001:
root@dance-with-linux:~#
```

اسم گروه Dance به with تغییر کرد.

مثلا برای تغییر جی‌آی‌دی گروه with می‌توان نوشت:



از 1001 به 2019 تغییر یافت.

اسکلت یک shell script

shell scriptهایی که می‌نویسیم باید جوری نوشته شده باشند که اگر چند سال بعد هم برگشتیم بفهمیم چی نوشتیم. برای همین یک اسکلت یا یک ساختار، (خارجکی script format) را تعریف می‌کنیم و از این به بعد از آن استفاده می‌کنیم:

1.shellای که این اسکریپت برای آن نوشته شده را تعریف کنید. مثلا: #!/bin/bash

این خط می‌گوید که این اسکریپت در bash اجرا می‌شود.

اگر نوشته باشیم: # ! /bin/corn

یعنی این اسکریپت در corn اجرا می‌شود.

2.کامنت بذارید.

همون که می‌گه یک کلام از مادر عروس، اونو اینجا بنویسین که این برنامه چی کار می‌کنه

#comments

3.تعریف متغیرها

بعد از اینکه کامنت‌ها و توضیحات را نوشتید، متغیرهای برنامه را بنویسید

مثلا می‌توان نوشت:

a= ‘my name is sahar’

a یک متغیر است.

4.کامندها

حالا وقت آن رسیده که کامندهایی که می‌خواهید اجرا کنید را بنویسید.

مثل echo، cp، grep و....

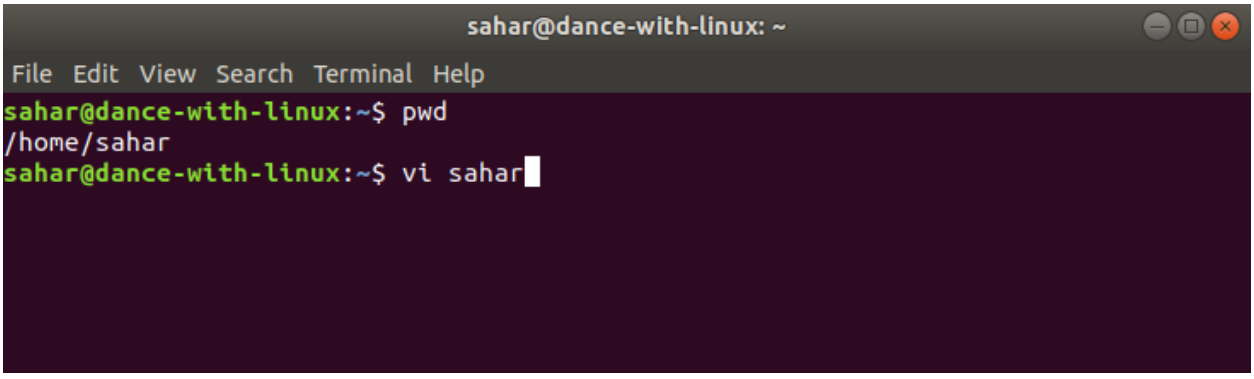
5.statements

حالا جملات یا همان statmentها را بنویسید (for, if, while ,)

موقع نوشتن statementها حتما حتما از tab استفاده کنید چون از طریق نشان می‌دهید که به فرض if از اینجا شروع شده و در اینجا تمام شده است.

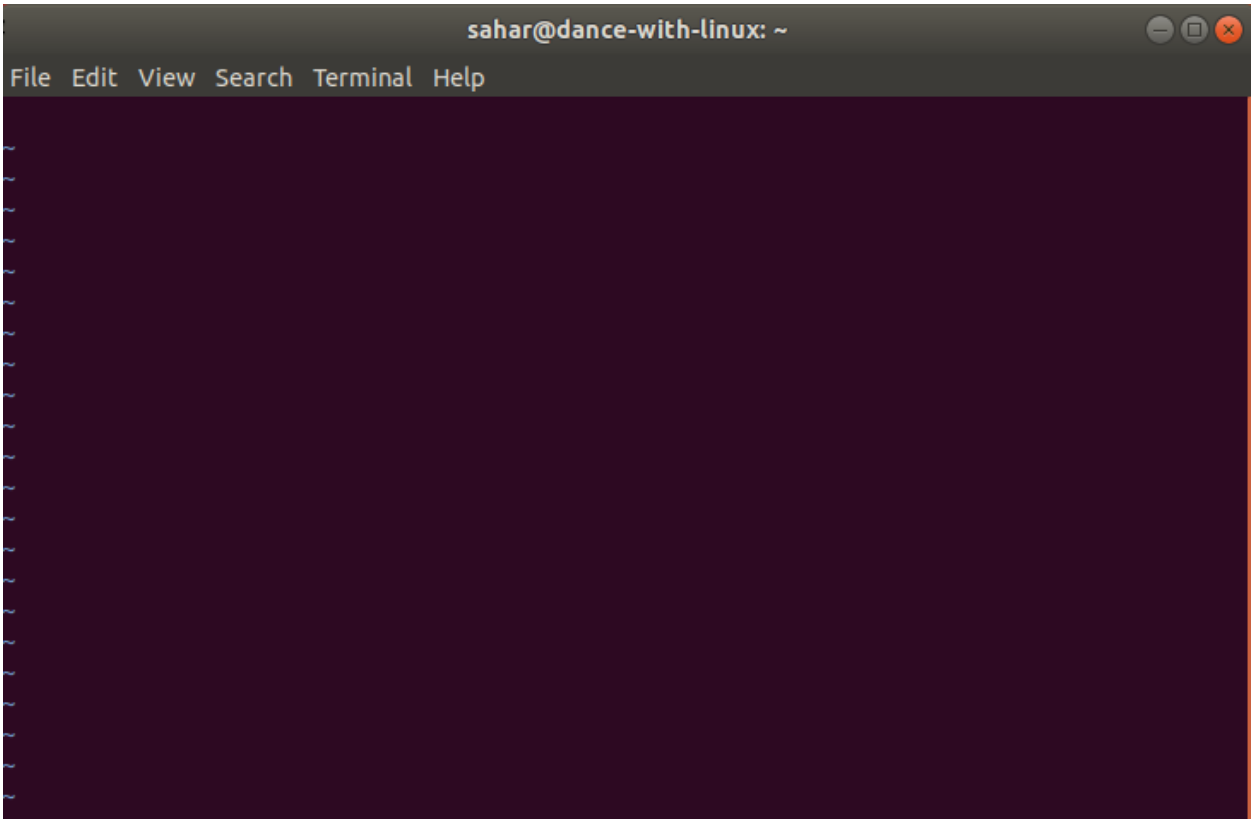
این 5 مرحله را در عمل ببینیم:

من از قبل یک فایل به اسم sahar ساختم، برای تغییر این فایل از کامند vi استفاده می‌کنم:



حتی اگر اون فایل رو از قبل نداشته باشین با vi می‌سازه و بعد تغییرش می‌ده، اما در کل می‌دونیم که با کامند touch می‌شه فایله رو ساخت.

وقتی Enter را بزنید، صفحه‌ی زیر نمایش داده می‌شود و شما می‌توانید نوشتن shell را شروع کنید:



نمی‌تونم بنویسم

ا رو بزن

در شکل زیر، گفتیم از چه شلی استفاده می‌کنیم و توضیحات را نوشتیم:

```
#!/bin/bash
# purpose: testing script format
# Date : 02/06/2019
# Modification: 02/06/2019
# be hengam shena mesle ye dasto pa cholofti
# be pa be masire dahan kose nayofti
~
~
~
~
~
```

در قدم بعدی، متغیر را نوشته و فراخوانی کردیم:

```
#!/bin/bash
# purpose: testing script format
# Date : 02/06/2019
# Modification: 02/06/2019
# be hengam shena mesle ye dasto pa cholofti
# be pa be masire dahan kose nayofti

a = 'My name is sahar'

echo $a
~
~
```

بعد کامندها را آوردیم مثلا دستور date

```
#!/bin/bash
# purpose: testing script format
# Date : 02/06/2019
# Modification: 02/06/2019
# be hengam shena mesle ye dasto pa cholofti
# be pa be masire dahan kose nayofti

a = 'My name is sahar'

echo $a

date
~
~
```

قدم بعدی آوردن statementها است، گفتیم برای اینجا حواستان باشد که یا از space یا tab استفاده کنید تا مشخص شود که این statement از کجا شروع شده و کجا تمام شده است:

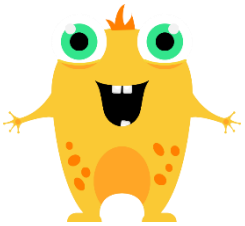
```
#!/bin/bash
# purpose: testing script format
# Date : 02/06/2019
# Modification: 02/06/2019
# be hengam shena mesle ye dasto pa cholofti
# be pa be masire dahan kose nayofti

a = 'My name is sahar'

echo $a

date

        if { $a -eq file}
        then echo this
        else
        echo that
        fi
~
~
```

با fi که آخر نوشته شده، در واقع statement را بسته. اینکه تمام این‌ها چیست و چه کاری انجام می‌دهد را در ادامه گفته می‌شود. هدف در اینجا ساختار یا اسکلت نوشتن یک shell script بود.

در آخر سر هم با تایپ :wq! فایل را ذخیره کرده و می‌بندیم:

```
~
~
:wq!
```

کار نکرد

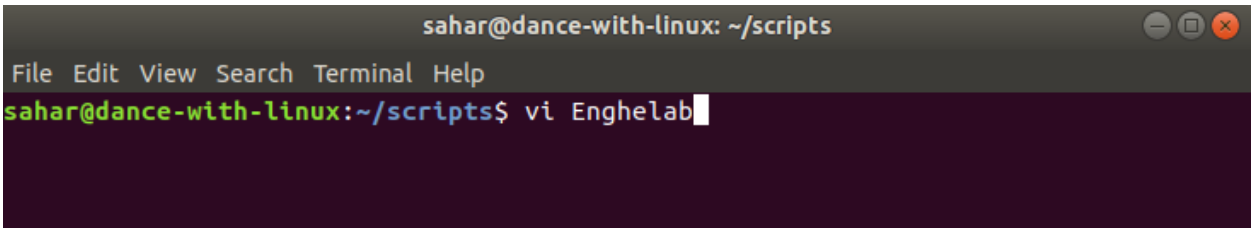


اگر زدین :wq! کار نکرد یه بار Esc رو بزنین دوباره امتحان

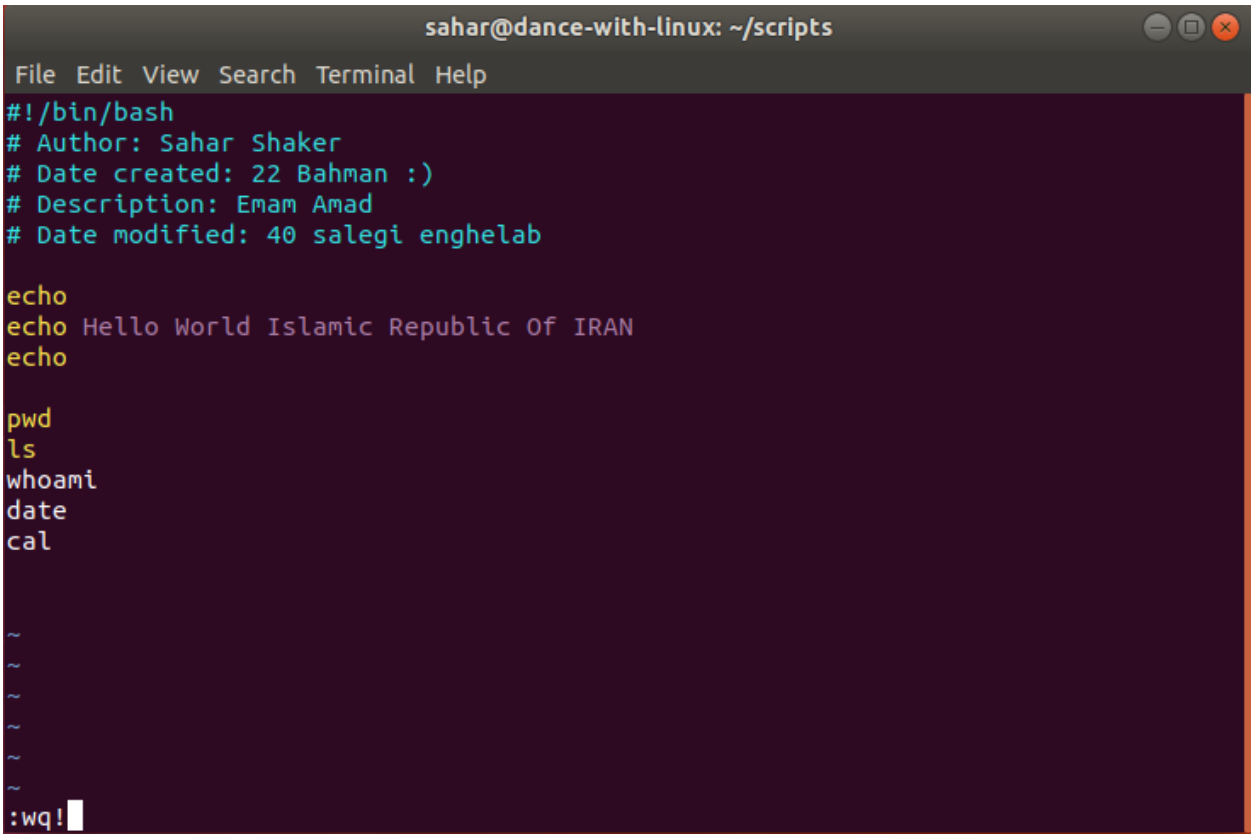
بریم سراغ Hello world

بیایید یک اسکریپت ساده‌ی دیگر بنویسم که Hello world را نشان دهد و چندین دستور ساده مثل cal و Date و... را داشته باشد:

گفتیم از قبل که تمام اسکریپت‌ها را در یک دایرکتوری، مثلا اینجا دایرکتوری scripts، ذخیره کنید. فایل انقلاب را در دایرکتوری scripts ساختیم:



گام‌هایی که گفته بودیم را یکی یکی اجرا می‌کنیم:



می‌شه سیاسیش نکنی؟ و انقدر سوتی ندی؟؟؟

برای اجرای اسکریپت، با Relative path یا Absolute path که قبلا بحث کردیم، اجرا می‌کنیم:

```
sahar@dance-with-linux: ~/scripts
File  Edit  View  Search  Terminal  Help
sahar@dance-with-linux:~/scripts$ vi Enghelab
sahar@dance-with-linux:~/scripts$ ./Enghelab
```

وقتی Enter را می‌زنیم با خطای زیر مواجه می‌شویم:

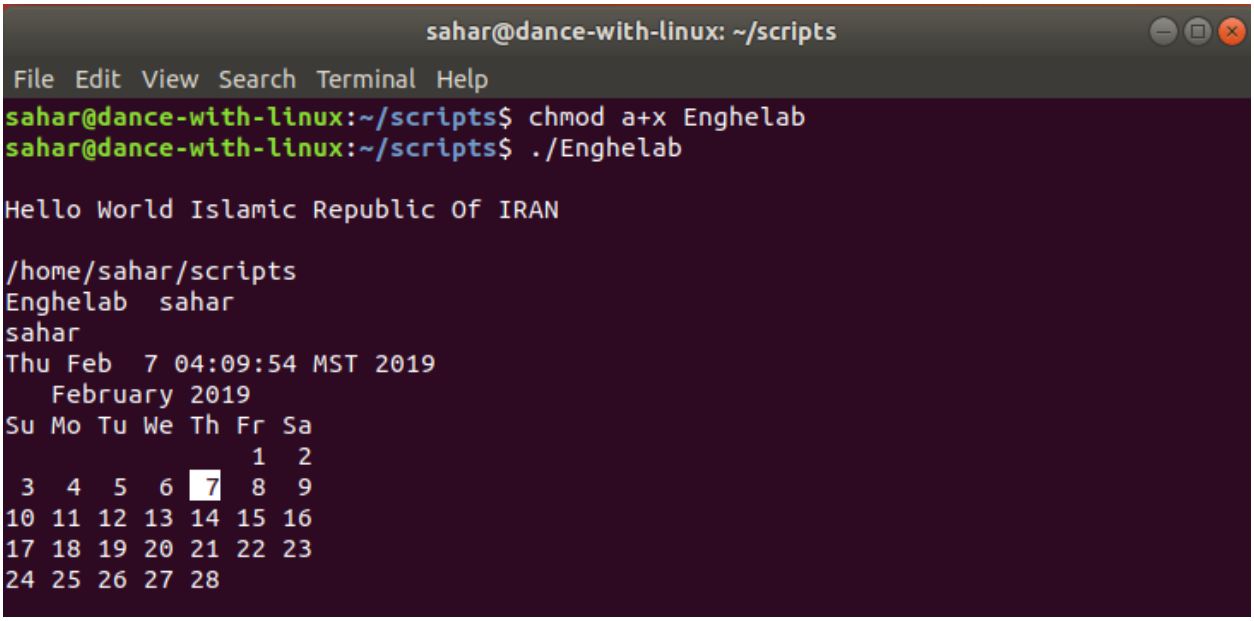
```
sahar@dance-with-linux:~/scripts$ ./Enghelab
bash: ./Enghelab: Permission denied
sahar@dance-with-linux:~/scripts$
```

دلیل چنین خطایی این است که اجازه‌ی "اجرا" ی این فایل را نداریم. در تصویر زیر می‌بینیم که تنها اجازه‌ی

Read و write را داریم:

```
sahar@dance-with-linux:~/scripts$ ls -l Enghelab
-rw-r--r-- 1 sahar sahar 206 Feb  7 03:58 Enghelab
sahar@dance-with-linux:~/scripts$
```

برای تغییر پرمیشن‌ها، روضه خواندیم که یا می‌توانید با chmod 777 fileName جلو برید یا از دستور chmod a+x fileName همه‌ی پرمیشین‌ها را بدهید؛ و بعد دوباره آدرس اسکریپت را بدهید:



در خروجی می‌بینید که یکی یکی دستوراتی که داده بودیم را اجرا کرده است.

تعریف متغیر

در ابتدایی‌ترین حالت اگر در نظر بگیریم، تعریف متغیرها بسیار ساده است. فرض کنید می‌خواهیم یکسری متغیر تعریف کنیم که این متغیرها دستوراتی را اجرا کنند.

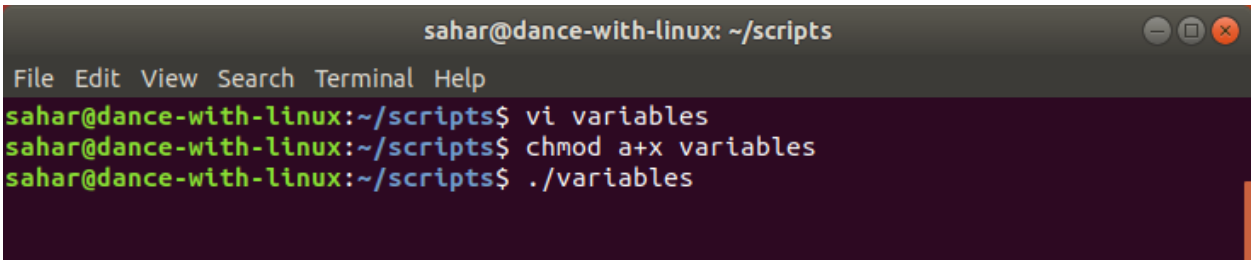
مثلا متغیر p را بگوییم که دستور pwd را اجرا کند:

```
sahar@dance-with-linux: ~/scripts
File  Edit  View  Search  Terminal  Help
#!/bin/bash

# Define variables

p= pwd
l= ls
c= cal 2018
d= date
w= whoami

echo Run variables tasks
echo

$p
$l
$c
$d
$w

~
~
~
~
:wq!
```

برای اینکه متغیرهایی که تعریف کردیم، ران کنیم با $ صداشون کردیم

همیشه بحثه پوله می‌دونم همیـــــــشـــــه

بعد از دادن پرمیشن‌ها هم اجرایش می‌کنیم:



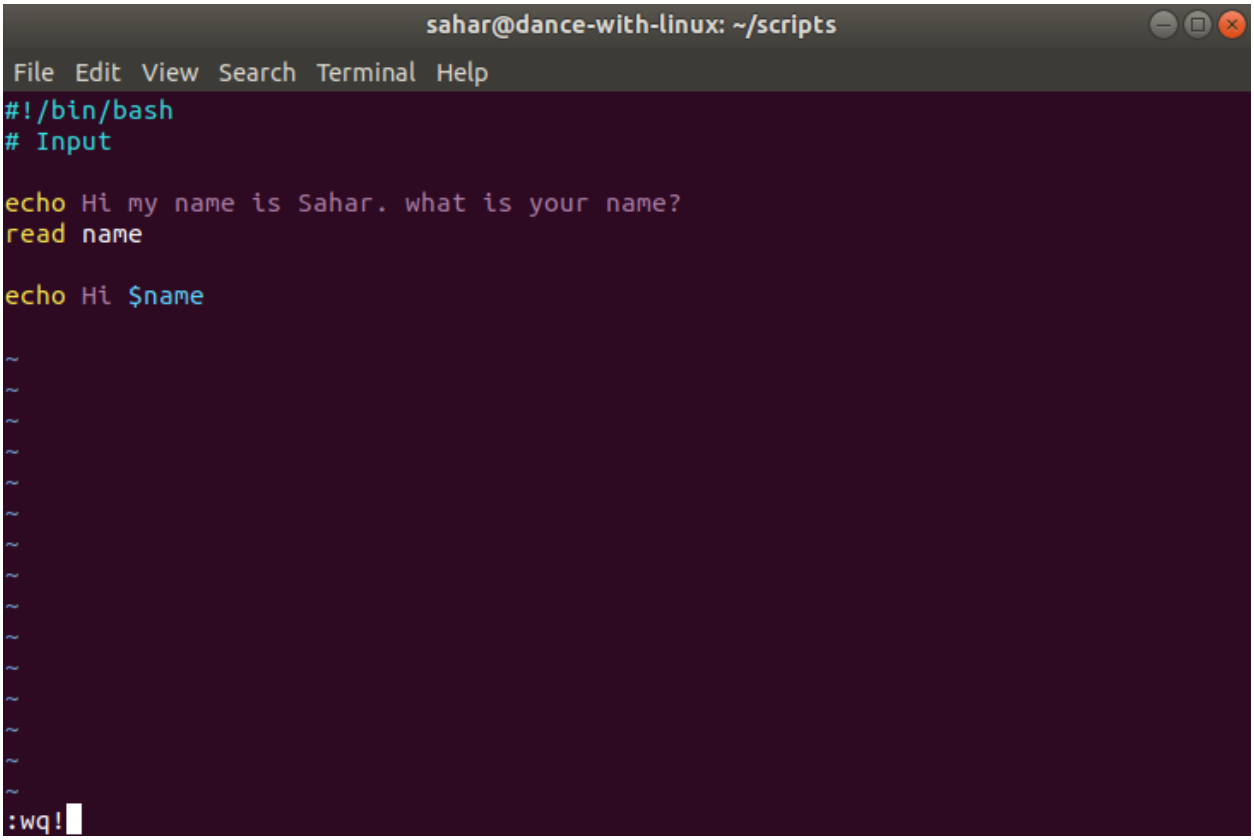
ورودی‌ها و خروجی‌ها

منظور از ورودی‌ها و خروجی‌ها، همان input/output است. گاهی اوقات نیاز است تا اسکریپتی که نوشتیم از کاربر ورودی بگیرد یا خروجی داشته باشد. در این حالت، اسکریپت می‌ایستد تا کاربر ورودی را بدهد و باقی اسکریپت را اجرا کند. یعنی تا کاربر اطلاعات را وارد نکرده است، اسکریپت ادامه نمی‌دهد.

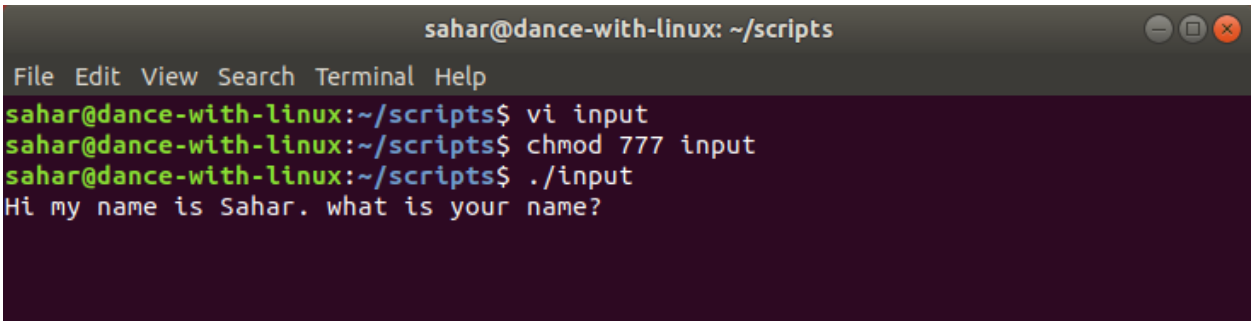
فقط توجه کنید که خیلی پیش می‌آید که اسکریپت ما، ورودی‌اش را از خروجی یک برنامه‌ی دیگر می‌گیرد. یعنی خیلی منظورمان از کاربر، آدمیزاد نیست هر چند کامپیوترها برای یک برنامه‌نویس بیشتر از هر آدم دیگر، آدم را درک می‌کنند!

در این حالت، بیشتر از پایپ | استفاده می‌شود.

در اینجا به ساده‌ترین حالتش اشاره می‌کنیم:



با دستور read از کاربر ورودی را می‌گیرد و با علامت $ ورودی را فراخوانی می‌کند، وقتی اسکریپت به قسمت read رسید، منتظر می‌ماند تا ورودی را بگیرد:



بعد از دادن ورودی:

```
sahar@dance-with-linux: ~/scripts
File  Edit  View  Search  Terminal  Help
sahar@dance-with-linux:~/scripts$ vi input
sahar@dance-with-linux:~/scripts$ chmod 777 input
sahar@dance-with-linux:~/scripts$ ./input
Hi my name is Sahar. what is your name?
Linux
Hi Linux
sahar@dance-with-linux:~/scripts$
```

مگه خودتْ خواهر مادر نْداری؛

خبر نامه جدا

بر بر مطا بر بر

سایت زیبا

صد فا از

رهه خته طایم

می بیم

کتا ب فصال م

مذ

مخصوص شما

Dance with Linux

Saharshaker.com

Update: 97/11/22