«به نام پروردگار»

گزارش پروژه
درس سیستم های عامل

استاد درس : دکتر نسترن طاهری جوان

تدریس یاران : مهندس کسرا مجلل – مهندس کسرا فرخی

نویسنده : محسن محمدیان

شماره دانشجویی : ۹۸۳۱۵۰۲

سوال ۱/۱)

برای نصب ابتدا باید با دستور زیر :

sudo apt install gcc libncurses5-dev dpkg-dev libssl-dev flex bison

ابزارهای لازم برای کامپایل linux kernel را نصب کنیم.

پس از آن به دایرکتوری موردنظر رفته و با دستور make menuconfig ابزار پیکربندی لینوکس را بالا می آوریم؛ که در آن یک منو با ویژگی‌ها و بخش‌های گوناگون برای شخصی سازی کرنل در برابر ما ظاهر می شود.

در قسمت پایین منو، ما چهار گزینه داریم (<select>, <Exit>, <Help>, <Save>, <Load>

که گزینه اول برای انتخاب آیتم، گزینه دوم برای خارج شدن، گزینه سوم برای save پیکربندی در فایل موردنظر و ادامه مراحل نصب؛ و گزینه چهارم برای آن است که اگر احیانا خواستیم یک پیکربندی ذخیره شده از گذشته را جایگزین کنیم؛ از این گزینه استفاده میکنیم. حتی می شود از این گزینه برای آنکه تنظیمات کرنل خود را بازیابی کنیم بجای آنکه آنرا ریستارت کنیم؛ استفاده کرد.(در هنگامی که جایی از تنظیمات را خراب کردیم و همه چیز را بهم ریختیم؛ می شود پیکربندی گذشته را که در فایل ".config" ذخیره شده را بازیابی کنیم.)

در این menu، ویژگی های و فیلدهای بسیاری وجود دارد که به تنظیمات لینوکس کرنل مربوط می شود و با کمک آنها می توانیم کرنل خود را باتوجه به نیازهای خود شخصی سازی کنیم.

این تنظیمات (مطابق شکل 1.1) به بخش های گوناگونی تقسیم می شود که هرکدام شامل یک زیرمجموعه از تنظیمات کرنل هست.

در ادامه به شرح برخی از آنها برای نمونه خواهیم پرداخت.

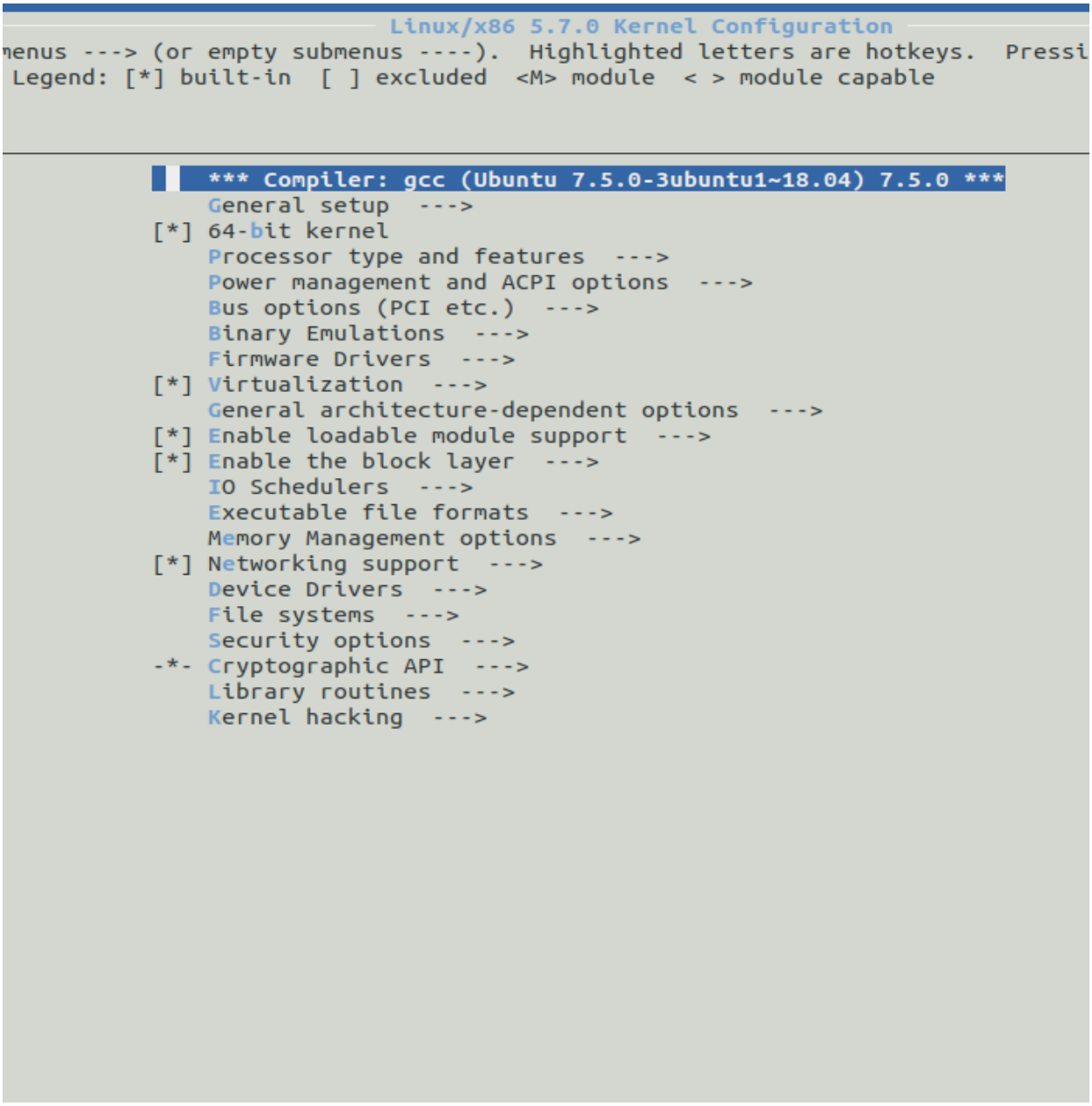
هرکدام از ویژگی های منو یا میتوانند built in شوند یا modularized شوند و یا ignore شوند. یهم چنین هرکدام از آیتم های منو، یا با نماد []، شروع می شوند یا <> و یا {}. که :

[] : نشان دهنده آن است که آن آیتم یا میتواند built in شود یا removed.

<> : نشان دهنده آن است که آن آیتم یا میتواند built in شود یا modularized یا removed.

و {} : نشان دهنده آن است که آن آیتم یا میتواند built in شود یا modularized (توسط آیتم های دیگر انتخاب شود)

داخل نمادهای بالا، یا * قرار می گیرد؛ یا M و یا whitespace که به ترتیب هرکدام بیانگر آن است که آیتم موردنظر یا built in شود؛ یا modularized و یا درنظر گرفته نشود (removed شود/exclude شود.)



شکل ۱,۱

آیتم های موجود در هر دسته معمولا از جنس Boolean هستند و وقتی گزینه help را روی هرکدام بزنیم یک صفحه باز می شود که درباره آن آیتم توضیح داده شده و سپس مشخص کرده type آن آیتم چیست(Boolean,tristate,...) و اینکه در کدام فایل و کدام خط برنامه تعریف شده؛ و چه مقادیری میتواند بگیرد و با چه symbol ای نمایش داده می شود.

آیتم General setup :

این آیتم خود مقدار زیادی attribute و فیلد برای تنظیمات general کرنل دارد.برای مثال می توان فیلد compile also drivers witch will not load را نام برد که با فعال کردن آن میتوان driver هایی را که حمایت سخت حمایت سخت افزار برای لود شدن یا استفاده شدن روی پلتفرم های گوناگون را ندارند می شود این اجازه را داد که از آنها استفاده شود.(بیشتر مورد استفاده developer ها قرار میگیرد)

یا local version – append to kernel release که یک string اضافه ای که ما مشخص می کنیم را به انتهای ورژن کرنل append می کند.

آیتم 64-bit kernel :

برای ساخت کرنل 64 بیتی (x86_64) Y میگیرد و برای ساخت کرنل 32 بیتی (i386)،N.

آیتم Processors type and features :

این آیتم نیز خود از آیتم های بسیاری تشکیل شده که برای مثال می توان از :

DMA memory allocation support—که به دستگاه هایی با کمتر از 32 بیت آدرس دهی، به اولین فضای آدرس دهی 16 مگابایتی اختصاص داده شوند.اگر دستگاهی برای چنین استفاده ای نداریم بهتر است این ویژگی را disable کنیم.—

یا Multi core scheduler support– که تصمیم گیری های CPU Scheduler را هنگام سر و کله زدن با تراشه های CPU چند هسته ای بهبود می بخشد.(در برخی جاها ممکن است این آیتم اندکی باعث افزایش overhead شود)— نام برد.

آیتم ACPI :

پشتیبانی رابط انرژی و پیکربندی پیشرفته (Advanced configuration and Power Interface) نیازمند یک پلتفرم (سخت افزار) سازگار با آن هست و همچنین حضور نرم افزار پیکربندی مستقیم OS و مدیریت انرژی (Power Management) را شامل می شود.

این option باعث توسعه ما توسط ACPI می شود؛ که Linux ACPI، یک جایگزین قوی برای پیکربندی میراث ها و رابط های مدیریت انرژی مهیا میکند.

این بخش آیتم هایی دارد که بعنوان driver معرفی شده اند؛ مانند : fan و button و processor و... . برای مثال درایور fan از fan های دستگاه های دیگر پشتیبانی میکند و به ما اجازی میدهد تا آنرا به

گونه ای پیکربندی کنیم که آیا برنامه های سطح کاربر، کنترل های ساده fan (مانند روشن و خاموش کردن) را بتوانند اجرا کنند یا خیر.

آیتم BUS Options :

برای پشتیبانی از ISA BUS در سیستم های مدرن هست.که در این بخش option ها و درایورهای دستگاه ها را برای انتخاب و پیکربندی در دسترس قرار می دهد.در سیستم های مدرن ISA BUS وجود ندارد و جایگزین آن PCI هست.

آیتم Binary Emulation :

برای مثال این بخش یک آیتم بنام IA32 Emulation دارد که با روشن کردن آن، برنامه های 32 بیتی میتوانند روی کرنل 64 بیتی اجرا شوند.

آیتم Virtualization :

این آیتم نیز خود از آیتم های بسیاری تشکیل شده است.محتوای کلی آن درباره میزبان شدن لینوکس ما و اجرا کردن سیستم عامل های دیگر (OS های مهمان) در ماشین های مجازی هست.

آیتم General architecture-dependent option :

این آیتم نیز خود از ویژگی ها گوناگونی تشکیل شده که یکی از آنها Oprofile system profiling هست؛ که یک سیستم profiling با گنجایش پروفایل کردن کل سیستم

(kernel, kernel modules, libraries, applications, …) هست.

آیتم Enable loadable module support :

ماژول های کرنل، تکه های کوچکی از کد هستند که میتوانند درهنگام اجرای کرنل در آن insert شوند بجای آنکه کاملا داخل آن ساخته شوند.(built in شوند) و از ابزار modprobe برای حذف و اضافه کردن آنها استفاده می شود.بیشتر برای option هایی استفاده می شود که به ندرت درهنگام بوت استفاده می شوند.

آیتم IO scheduler :

این آیتم نیز خود از ویژگی ها و آیتم های گوناگونی چون The Kyber I/O scheduler هست؛ که یک scheduler با overhead اندک برای سیاست چند صفی و سایر دستگاه های سریع هست. باتوجه تاخیرهای هدف برای خواندن و نوشتن های همزمان، برای رسیدن به صف، عمق آنرا تنظیم خواهد کرد.

آیتم Memory management Option :

این option نیز خود از آیتم های زیادی تشکیل شده که به ما اجازه می دهد تا برخی از روش هایی که لینوکس با آن حافظه خود را مدیریت میکند را تغییر دهید.بیشتر کاربران دراینجا تنها یک option دارند که توسط پیکربندی معماری (architecture configuration) انتخاب شده است.

آیتم library routins :

تمام متغیرهای کاربردی RAID6 PQ را روی init بررسی می کند و سریع ترین آنرا انتخاب می کند.

توضیحات بیشتر به همراه فیلدهای موجود در هر دسته ی اصلی option یا آیتم در منو پیکربندی در یک پوشه جداگانه به نام menuConfig قرار داده شده است.

پس از آنکه پیکربندی مطابق میل خود را انجام دادیم و ویژگی های موردنیاز را انتخاب کردیم یا تغییر دادیم، مراحل نصب را ادامه می دهیم به این صورت که :

ابتدا تنظیمات اعمال شده را در یک فایل(مثلا ".config") ذخیره میکنیم و سپس Exit میکنیم.

سپس با دستور

make –j 5 KDEB_PKGVERSION=1.CustomName deb-pkg اقدام به کامپایل کردن کرنل میکنیم؛ که CustomName می تواند برای هرکسی اسم موردنظر خودش باشد(مثلا اسم سیستم لینوکس خود شخص)

سپس با دستور sudo dpkg - i ../linux*.deb، کرنل را نصب می کنیم.پس از از آنکه همه چیز به خوبی پیش رفت، نیاز به Reboot شدن سیستم می شود و پس از ریستارت با دستور uname –r میتوانیم ورژن کرنل خود را مشاهده کنیم.

همچنین برای uninstall کردن آن نیز میتوانیم از دستور:

Sudo apt purge linux-image-versionOfKernelLinux linux-image-versionOfKernelLinux-dbg

استفاده کنیم.

• what is Kernel?

هر سیستم عاملی چه لینوکس باشد، چه ویندوز، چه Mac، چه اندروید، یک برنامه هسته ای درون خود دارد که نقش یک رئیس را در تمام سیستم بازی می کند. درواقع یک کرنل چیزی نیست جز یک برنامه کامپیوتری که همه چیز را در سیستم کنترل می کند.

این برنامه پس از bootloader، اولین برنامه ای است که در سیستم لود می شود؛ سپس تمامی ارتباطات و حرف زدن های بین نرم افزار و سخت افزار را کنترل میکند.هنگامی که ما یک برنامه را اجرا میکنیم، رابط کاربری یک درخواست به کرنل می فرستد؛ سپس کرنل یک درخواست به CPU و

memory می فرستد تا انرژی پردازشی و حافظه و چیزهای دیگر را دراختیار قرار دهد تا برنامه بتواند در سمت front به راحتی اجرا شود.

درواقع کرنل را می شود نوعی مترجم تصور کرد که درخواست های ورودی/خروجی از نرم افزار را به مجموعه ای از دستورالعمل ها برای CPU و GPU تبدیل می کند. به بیان ساده کرنل یک لایه بین سخت افزار و نرم افزار هست.

- **what is user mode and kernel mode?**

از آنجایی که منابع سخت افزاری و نرم افزاری میان سیستم عامل و کاربرانش به اشتراک گذاشته می شود ، یک سیستم عامل با طراحی خوب باید که یک برنامه مخرب باعث اختلال در عملکرد برنامه های دیگر یا خود سیستم عامل نمیشود. برای اینکه مطمئن شویم سیستم به درستی کار میکند باید بین کد کاربر و کد سیستم عامل وهمچنین حالت های اجرا تمایز قائل شویم به یک پشتیبانی سخت افزاری نیاز داریم که این کار با اضافه کردن یک بیت به سخت افزار انجام می شود.

دست کم در سیستم دو حالت (mode) وجود دارد :

یکی حالت کرنل (system mode , superviser mode)و دیگری کاربر(user mode) بیت اضافه شده به سخت افزار (bit mode) هنگامی که صفر باشد در حالت کرنل هستیم و هنگامی که یک باشد در حالت کاربر. با این بیت ما میتوانیم بین task ای که در سمت برنامه کاربر و task ای که در سمت سیستم عامل انجام می شود ، تمایز قائل شویم .

در زمان بوت سیستم سخت افزار در حالت کرنل(kernel mode) هست هنگامی که سیستم عامل بالا آمد و برنامه کاربر را اجرا کرد به user mode می رود. هر هنگام که trap یا وقفه ایجاد میشود سخت افزار از حالت کاربر به کرنل می رود(mode bit =0). درواقع هر زمانی که سیستم عامل کنترل کامپیوتر را برعهده می گیرد سخت افزار در حالت کرنل است و هرزمان که برنامه کاربر cpu را مشغول کرده سخت افزار در حالت کاربر است.

و درنهایت هنگامی که کنترل cpu دست سیستم عامل است ، برای برگشتن به حالت کاربر همیشه سیستم حالت سخت افزار را قبل از دادن کنترل به برنامه کاربر به user mode تغییر میدهد.

دو حالته بودن عملگر به ما این مفهموم را می رساند که میتوانیم ازسیستم عامل در برابر کاربران تهدید آمیز محافظت کنیم.

پس از مزایای آن میتوان گفت که میتوانیم بعضی از دستور های ماشین را که میتوانند سبب آسیب به ماشین شوند به عنوان دستورهای ابتدایی (privileged instructions) تعریف کنیم. سخت

افزار اجازه میدهد که دستورات ابتدایی تنها در مود کرنل اجرا شوند و اگر کاربر تلاش کند که آنرا در حالت کاربر اجرا کند ، سخت افزار آنرا اجرا نخواهد کرد و با آن مانند یک trap یا دستور illegal برخورد خواهد کرد.(برای مثال از privileged instructions میتوان خود دستور تغییر مود به حالت کرنل ، کنترل های i/o ، مدیریت زمان و... را نام برد.)

همچنین مفهموم حالت ها میتواند بیش از دوتا باشد برای مثال پردازنده های intel چهار حلقه ی محافظ جداگانه دارند که 0 حالت کرنل و حلقه 3 حالت کاربر است (حلقه 1 و2 برای سایر خدمات سیستم عامل)

یا سیستم های ARMv8 که هفت mode دارند برای مدیریت ماشین های مجازی ونشان از اینکه vm آنها چه زمانی تحت کنترل سیستم از است ، پردازنده آنها حالت های مختلفی دارد. vm امتیاز های بیشتری نسبت به حالت کاربر دارد اما امتیاز های آن از حالت کرنل کتر است.

- **what happens if ext4 driver is removed?**

بدون پیاده سازی صریح ext4 filesystem و enable کردن آن، فرمت filesystem روی دیسک به همان گونه گذشته (سازگار با فرمت قبلی) باقی می ماند.

یا به عبارتی کاملا backward compatible می شود.

سوال ۱/۲)

برای پیاده سازی این قسمت به دو فایل نیاز بود، که در یکی از آنها کد ماژول مورد نظر (hello-1) نوشته شد(که با زبان c هست و پسوند آن c. می باشد) و دیگری Makefile بود که در آن چند خط کد شامل ، یک شیء از فایل ماژول و target های clean و all وجود داشت(که نوع آن txt می باشد).

برای کامپایل کردن ماژول خود پس از ساختن فایل Makefile، لازم هست تا با رفتن به دایرکتوری ای که این دو فایل در آن قرار دارند، دستور make را بزنیم تا کامپایل انجام شود.با دستور

modinfo moduleName.ko نیز می توانیم اطلاعات مربوط به ماژول را مشاهده کنیم.حال وقت اضافه کردن ماژولی که ساخته ایم به کرنل هست؛ که این کار با دستور **sudo insmod moduleName.ko** انجام می شود(که پسوند ko در نام فایل بیانگر kernel object هست).

تمامی ماژول ها در کرنل، در دایرکتوری `/proc/modules` قرار دارند که برای اینکه اطمینان حاصل کنیم ماژول ما اضافه شده یا نه می توانیم فایل modules را چک کنیم.

همچنین با دستور **sudo rmmod moduleName**، می توانیم ماژول اضافه شده را از کرنل حذف کنیم.برای اینکه هم ببینیم آیا ماژول ما به درستی add یا remove می شود و توابع ()init_module و

()cleanup_module به درستی کا می کنند و رشته ای که در این دو تابع با printk، log کرده ایم را ببینیم می توانیم در یک پنجره ترمینال دیگر در روت، دستور tail –f /var/log/kern.log را بزنیم.
در این صورت وقتی ماژول را add می کنیم پیام Hello World یا GoodBye World را در پنجره دیگر ترمینال می بینیم؛ یا میتوانیم با مراجعه به /var/log/messages از log شدن پیغام ها در سیستم logfile خود مطمئن شویم.

برای آنکه هم که مجبور نباشیم از نام های اصلی ()init_module و ()cleanup_module استفاده کنیم، می توانیم با include کردن کتابخانه linux/init.h و قرار دادن __init و __exit پیش از توابع اصلی بالا، نام دلخواه خود را قرار دهیم.

اما حتما باید در آخر از ماکروهای ()module_init و()module_exit پس از تعریف توابع init و cleanup_module با نام دلخواه خود استفاده کنیم و به عنوان ورودی نام تابع را به ماکروهای بالا بدهیم.

- **What would happen if there was no driver module?**

یک نوع از انواع ماژول ها device driver هست که به کرنل اجازه آنرا می دهد تا به سخت افزار متصل به سیستم دسترسی داشته باشد. بدون ماژول ها ما باید یک کرنل یکپارچه را بسازیم و functionality های جدید را مستقیما به kernel image اضافه کنیم؛ همچنین علاوه بر بزرگ شدن کرنل این مضرت را دارد که باعث می شود هرزمان ما میخواهیم یک functionality تازه اضافه کنیم مجبوریم آنرا reboot و rebuild کنیم.

- **what is the difference between insmod and modeprobe?**

وقتی module ای در سیستم نیست kmod ، و modeprobe را اجرا می‌کند تا module موردنظر را لود کند.
Modeprobe یک string را به یکی از دو فرم نام module یا identifier آن می گیرد.وقتی با یک generic identifier سرگرم هست، ابتدا در فایل /etc/modprobe.conf به دنبال آن می‌گردد و اگر یک خط مانند alias char-major-10-30 softdog پیدا کرد
متوجه می‌شود که generic identifier به softdog مثلاً اشاره می‌کرده است.سپس modeprobe در فایل /lib/modules/version/modules. dep/ می‌بیند که ماژول به ماژول های دیگری وابسته هست و آیا نیاز هست پیش از لود آن، ماژول های دیگری لود شوند یا خیر.
درآخر modeprobe از insmod برای لود کردن ماژول های پیش نیاز و ماژول موردنظر استفاده می کند.
تفاوت modeprobe و insmod آن است که insmod نسبت به مکان ماژول ها به نسبت گنگ است و باید آدرس کامل ماژول ها به آن داده شود؛ ولی modeprobe، از مکان پیش‌فرض ماژول ها آگاه هست و می‌داند یک ماژول به چه ماژول هایی وابسته است و درنتیجه از اینکه ماژول ها به چه ترتیبی باید لود شوند، آگاه است.

چیزی که در بالا واضح است، آن است که insmod به یک pathname کامل برای هر ماژول نیاز دارد و همچنین نیاز دارد تا ماژول ها را به ترتیب صحیح وارد کنیم(insert کنیم) اما modeprobe یک نام ماژول را می‌گیرد و تمام چیزی را که نیاز دارد با پارس کردن /lib/modules/version/modules.dep امی فهمد.

- **what are init_module and cleanup_module?**

هر ماژول کرنل دست کم دو تابع (function) دارد؛ یکی تابع آغاز (initialization) که تابع init_module نامیده می‌شود و هنگامی فراخوانی می‌شود که ماژول در کرنل (insert) insmod می شود؛ و دومی نیز تابع پایان (cleanup) هست که cleanup_module نامیده می‌شود و درست پیش از آنکه ماژول (remove) rmmod شود، فراخوانی می شود.
پس از کرنل 2.3.13 ، میتوان از هر نامی برای توابع init و cleanup استفاده کرد.
عموما ()init_module، یا یک کنترل کننده (handler) را برای چیزهایی، با کرنل ثبت می‌کند؛ یا کد خود را جایگزین یکی از توابع کرنل می کند. حال cleanup_module قرار هست که هرکاری که ()init_module کرده را عقبگرد کند؛ بنابراین ماژول موردنظر می‌تواند به صورت امن unload شود.

سوال ۱/۳)

نتیجه ی قسمت اول سوال :

```
mohssenmhd@mohssen:~$ tail -f var/log/kern.log
tail: cannot open 'var/log/kern.log' for reading: No such file or directory
tail: no files remaining
mohssenmhd@mohssen:~$ tail -f /var/log/kern.log
Jun 26 19:48:20 mohssen NetworkManager[689]: <info>  [1593184700.2175]   gateway
 10.0.2.2
Jun 26 19:48:20 mohssen NetworkManager[689]: <info>  [1593184700.2178]   server
identifier 10.0.2.2
Jun 26 19:48:20 mohssen NetworkManager[689]: <info>  [1593184700.2182]   lease t
ime 86400
Jun 26 19:48:20 mohssen NetworkManager[689]: <info>  [1593184700.2186]   nameser
ver '8.8.8.8'
Jun 26 19:48:20 mohssen NetworkManager[689]: <info>  [1593184700.2189]   nameser
ver '8.8.4.4'
Jun 26 19:48:20 mohssen NetworkManager[689]: <info>  [1593184700.2191]   nameser
ver '192.168.1.1'
Jun 26 19:48:20 mohssen NetworkManager[689]: <info>  [1593184700.2193]   domain
name 'Home'
Jun 26 19:48:20 mohssen NetworkManager[689]: <info>  [1593184700.2195] dhcp4 (en
p0s3): state changed unknown -> bound
Jun 26 19:48:20 mohssen NetworkManager[689]: <info>  [1593184700.2236] dns-mgr:
Writing DNS information to /sbin/resolvconf
Jun 26 20:06:41 mohssen gnome-session-binary[1605]: Entering running state
Jun 26 20:24:16 mohssen kernel: [ 1833.330753] hello_1: loading out-of-tree modu
le taints kernel.
Jun 26 20:24:16 mohssen kernel: [ 1833.330758] hello_1: module license 'unspecif
ied' taints kernel.
Jun 26 20:24:16 mohssen kernel: [ 1833.330759] Disabling lock debugging due to k
ernel taint
Jun 26 20:24:16 mohssen kernel: [ 1833.330803] hello_1: module verification fail
ed: signature and/or required key missing - tainting kernel
Jun 26 20:24:16 mohssen kernel: [ 1833.331369] Hello world 1.
```

چون نتیجه ی قسمت دوم سوال ۱/۳ حجم زیادی می گرفت، به صورت یک فایل txt با نام 1_3.txt پیوست شده است.

• Part 2 – Scripts

سوال ۲/۱)

1. What is the command to create three directories that dir3 is a sub directory of dir2 and dir2 is a sub directory of dir1?

یا می توانیم در دایرکتوری موردنظر که می خواهیم این سه دایرکتوری تو در تو را بسازیم سه بار به طور متوالی از دستورات cd و mkdir استفاده کنیم؛ یا فقط از دستور زیر استفاده کنیم :

mkdir –p ~/Desktop/dir1/dir2/dir3

(برای مثال من این سه دایرکتوری را، در دسکتاپ ساختم. فولدر ایجاد شده پیوست شده)

که p– مشخص می کند اگر main directory موردنظرساخته نشده بود، آنرا هم ایجاد کند.

2. Use cat to create a file which has the following content: one-two-three-four-five

با دستور cat > filename.txt این کار انجام می شود.سپس محتوای ورودی(one-two-three-four-five) را وارد کرده و ctrl+d میکنیم. (فایل پاسخ در پوشه پیوست شده)

3. Can you access the root directory and why?

خیر، دلیل آن نیز این هست که اگر ما بتوانیم به روت دسترسی داشته باشیم و بخواهیم فایلی را در روت تغییر دهیم یا آنرا حذف کنیم، ممکن هست کل سیستم عامل بهم بریزد و به مشکل بخورد.

لازم هست به این نکته هم اشاره کرد که پس از استفاده از دستور cat و ساخت فایل تا به نوعی انگار ما یک ادیتور را اجرا کرده ایم که در صورتی این ادیتور بسته می شود که از ctrl+d استفاده کنیم و به آن بفمهانیم که به انتهای فایل رسیده ایم.

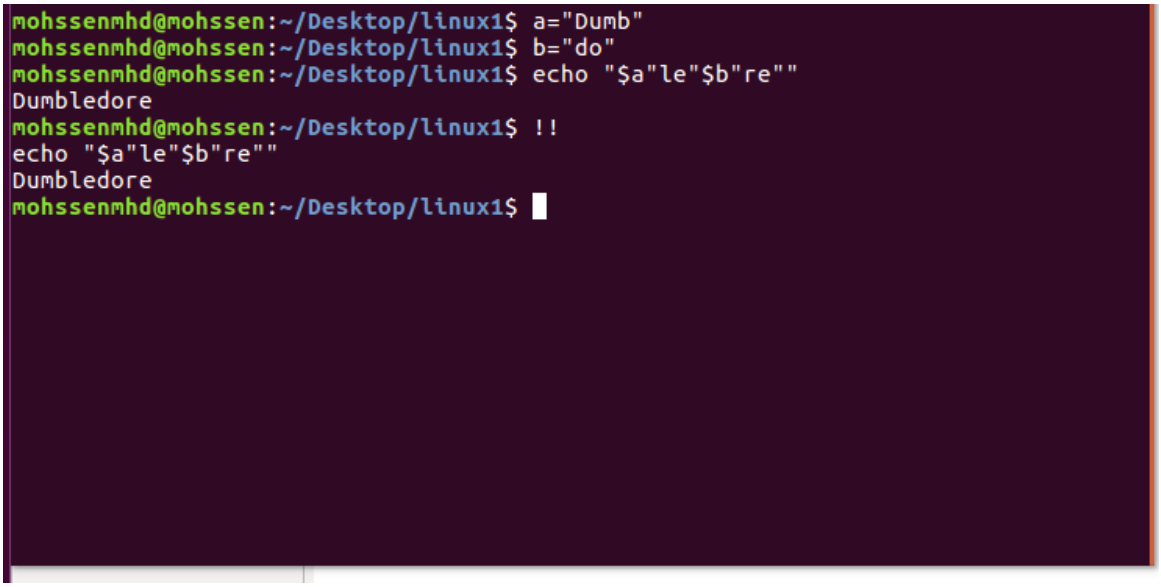
4. Where is the cat command stored?

در آدرس /usr/bin ذخیره شده است.

5. Create two variables, first one with the value Dumb, and the second with the value do, then use echo to printout Dumbledore.

این کار به دو روش قابل انجام است.

روش اول آن هست که دستورات را در ترمینال وارد کرده و متغیرها را تعریف کنیم و درنهایت با دستور echo آنها را چاپ کنیم. که در زیر نمونه ی آن آمده است :



روش دوم آن هست که کد مربوط به تعریف متغیرها و concat آنها با رشته ها را در یک فایل ذخیره کنیم و سپس با دستور آنرا اجرا کنیم. (فایل کد موردنظر sh. در پوشه ی پروژه قرار داده شده است)

6. Using to character, execute the previous command again.

با کاراکتر !! می توان دستور قبلی را دوباره اجرا کرد.

7. How can we store the users of the bash in a sorted list with the name bashusers.txt?

با دستور :

grep bash etc/passwd | cut –d: -f1 sort > bashusers.txt

این کار انجام می شود. که ابتدا با استفاده از دستو (فیلتر) grep، لیست userهای bash را (آنهایی که loging shell آنها در /bin/bashهست) از دایرکتوری etc/passwd در می آوریم یا به عبارتی کاربران bash را فیلتر می کنیم؛ سپس با دستور pipe یا (|) ، این لیست را به دستور یا فیلتر cut پاس می دهیم. در دستور cut ما با f1– تنها ستون اول فایل passwd را می گیریم که همان user name می باشد. پس تا اینجا ما لیست نام کاربری تمامی کاربران Bash را داریم؛ سپس این لیست را به دستور sort پاس میدهیم تا لیست را براساس نام کاربران مرتب کرده و در فایل bashusers.txt ذخیره کند.

8. How can we write a command to execute two commands with a sequence?

اگر دستورات به هم مرتبط نباشند و اجرای هریک وابسته به دیگری نباشد، می توان در ترمینال، از دستور

command1 ; command2 استفاده کرد؛ که به ترتیب command1 دستور اول و command2 دستور دوم هست که باید اجرا شود.

اگر دستورات به هم مرتبط باشند و اجرای هریک وابسته به دیگری باشد، می توان از دستور

command1 && command2 ، استفاده کرد. که اگر command1 به درستی اجرا شود، command2 هم پس از آن اجرا می شود.

در هردو روش بالا ابتدا دستور سمت چپ اجرا و سپس دستور سمت راست اجرا می شود.

سوال ۳٫۱)

در این قسمت حفاظت از ناحیه بحرانی برپایه FIFO و روش ticketlock پیاده سازی شده است.

که سه مدل کد برای این قسمت پروژه زده شده (در فولدری با نام ticketlock به فایل زیپ پروژه پیوست شده) که همه ی آنها به درستی کار کرده و جواب می دهند و درنهایت یکی از آنها بعنوان کد برگزیده در xv6 پیاده شده است.

نتیجه ی حاصل از از اجرای فایل تست برای این قسمت در زیر آمده است. که مقداری که هربار چاپ می شود دارد در واقع متغیر موجود در ticketlockTest system call را یکی اضافه می کند و با cprintf چاپش می کند. که با استفاده از aquire_ticketLock و release_ticketLock در هر زمان تنها یک فرآیند می تواند به آن دسترسی داشته و آنرا یکی اضافه و مقدارش را چاپ کند.

```
$ ticketLockTest
cchild adding to shared counter
***1***
hicld adding to shared counter
***2***
hild adding to shared counter
***3***
child adding tchild adding to shared counter
***4***
child adding to child adding to shared counter
***5***
child adding to shared counter
```

6

o shared counter

7

shared counter

8

child adding to child adding to shared counter

9

shared counter

10

child adding to shared counter

11

سوال ۳٫۲)

نتایج مرتبط با این قسمت :

هر نتیجه از دیگری با خطی از کاراکتر (*) جدا شده است.

enter pattern for readers/writers test

1110001

child adding to shared counter 1

writer added to shared counter

child adding tchild adding to shared counter 0

reader read from shared counter: 1

cohild adding to shared counter 1

writer added to shared counter

shared counter 0

reader crhild adding to shared counter 0

reader read from shared counter: 2

ead from shared counter: 2

child adding to shared counter 1

writer added to shared counter

user program finished

last value of shared counter : 3

---

enter pattern for readers/writers test

1100010

child adding to shared counter 1

writer added to shared counter

cchild adding to shared counter 0

reader read from shared counter: 1

hichild adding to shared counter 0

reader read from shared counter: 1

ld adding to shared counter 0

reader read from shared counter: 1

child adding to shared counter 1

writer added to shared counter

child adding to shared counter 0

reader read from shared counter: 2

user program finished

last value of shared counter : 2

---

enter pattern for readers/writers test

1001101

child adding to shared counter 0

reader read from shared counter: 0

child adding to shared counter 0

reader read from shared counter: 0

child adding to shared counter 1

writer added to shared counter

ccchild adding to shared counter 0

reader read from shared counter: 1

hilhild adding to shared counter 1

writer added to shared counter

d adding to shared counter 1

writer added to shared counter

user program finished

last value of shared counter : 3

---

enter pattern for readers/writers test

110101010

child adding to shared counter 1

writer added to shared counter

child adding to shared counter 0

reader read from shared counter: 1

cchild adding to shared counter 0

reader read from shared countehildr: 1

child adding adtco shared counter 1

writer added to shared counter

child adding to shared counter 1

writer added to shared counter

ding to shared counter 1

hwriter added to shared counter

ild adding to shared counter 0

child adding to shared counter 0

reader read from shared counter: 4

reader read from shared counter: 4

user program finished

last value of shared counter : 4