



Pontifícia Universidade Católica de Campinas
Faculdade de Engenharia de Computação -
FECOMP

Sistemas Operacionais B – Relatório Atividade 1
Compilação de um Kernel Linux

Beatriz Morelatto Lorente RA: 18071597

Cesar Marrote Manzano RA: 18051755

Fabricio Silva Cardoso RA: 18023481

Pedro Ignácio Trevisan RA: 18016568

Sumário

1.Introdução.....	3
2.Principais passos da compilação de um kernel Linux.....	4
3.Conclusão.....	10

Introdução

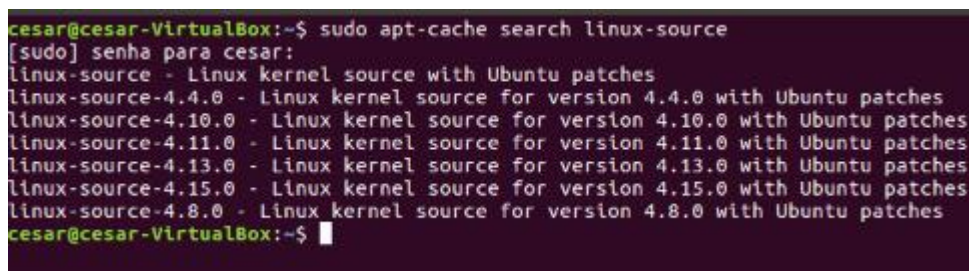
O experimento tem como objetivo familiarizar-se com todas as etapas de compilação de um kernel. Foi experimentado todo o processo de instalação, configuração e compilação do kernel. Foi necessário compreender cada fase para obtermos o resultado desejado, ou seja, com instalar um kernel, o que podemos desativar em um kernel Linux e como fazer sua compilação. Ao final do experimento é esperado que a equipe obtivesse um kernel que pudesse ser compilado rapidamente, com no máximo 30 minutos.

Principais passos da compilação de um kernel Linux

Passo a passo para a compilação de um kernel

Para a compilação de um kernel é necessário ter em mente que dados da sua máquina podem ser corrompidos, sua máquina pode não rodar mais, uma vez que ocorreu algum erro no kernel. Portanto é recomendado o uso de uma máquina virtual, ao invés de instalar o sistema Linux em sua máquina. Para o experimento foi utilizado o Ubuntu 16.04.

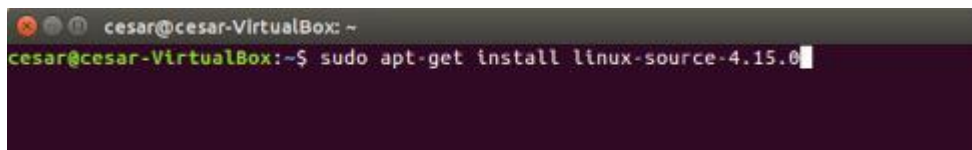
Para iniciarmos o processo de compilação é necessário baixar uma versão do kernel. Para isso, foi usado o comando `'sudo apt-cache search linux-source'`, no terminal do ubuntu, como mostrado na imagem abaixo.



```
cesar@cesar-VirtualBox:~$ sudo apt-cache search linux-source
[sudo] senha para cesar:
linux-source - Linux kernel source with Ubuntu patches
linux-source-4.4.0 - Linux kernel source for version 4.4.0 with Ubuntu patches
linux-source-4.10.0 - Linux kernel source for version 4.10.0 with Ubuntu patches
linux-source-4.11.0 - Linux kernel source for version 4.11.0 with Ubuntu patches
linux-source-4.13.0 - Linux kernel source for version 4.13.0 with Ubuntu patches
linux-source-4.15.0 - Linux kernel source for version 4.15.0 with Ubuntu patches
linux-source-4.8.0 - Linux kernel source for version 4.8.0 with Ubuntu patches
cesar@cesar-VirtualBox:~$
```

Figura 1 - Print mostrando o comando para procurar uma versão do kernel linux

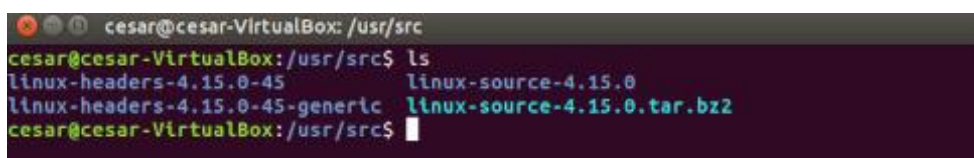
Ao executar o comando o terminal mostra várias opções disponíveis de kernel para a instalação. Para o experimento foi utilizado a versão 4.15.0. Para o arquivo ser baixado basta digitar o comando `'sudo apt-get install linux-source-4.15.0'` (se quiser baixar outra versão apenas troque os números da versão do linux).



```
cesar@cesar-VirtualBox: ~
cesar@cesar-VirtualBox:~$ sudo apt-get install linux-source-4.15.0
```

Figura 2 - Print mostrando o comando para instalar a versão desejada do kernel

A primeira parte para a compilação do kernel está feita. O download do arquivo se encontra na pasta `'usr/src'`. Basta navegar pelo terminal para acessá-la. Ao chegar na pasta vamos encontrar uma variedade de arquivos, como mostrado abaixo.

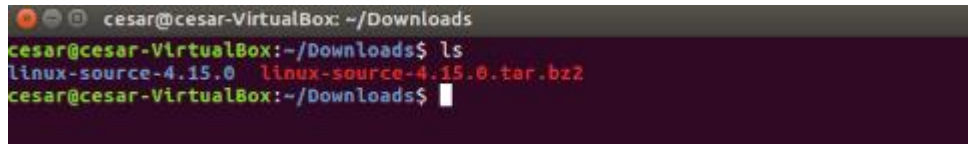


```
cesar@cesar-VirtualBox: /usr/src
cesar@cesar-VirtualBox:/usr/src$ ls
linux-headers-4.15.0-45          linux-source-4.15.0
linux-headers-4.15.0-45-generic linux-source-4.15.0.tar.bz2
cesar@cesar-VirtualBox:/usr/src$
```

Figura 3 - Print mostrando os arquivos da pasta `'usr/src'`

O arquivo desejado é o `'linux-source-4.15.0.tar.bz2'`. É recomendado que você copie esse arquivo para uma pasta separada, assim evitando qualquer tipo de conflito com os outros arquivos da pasta atual. Para copiar um arquivo digite o comando `'cp linux-source-4.15.0.tar.bz2 caminho-da-pasta-desejada'`.

Com isso feito é necessário descompactar o arquivo para podermos manipulá-lo. Para descompactar digite o comando `'tar xjpf linux-source-4.15.0.tar.bz2'`. Espere a descompactação terminar e você terá um resultado como na imagem abaixo.



```
cesar@cesar-VirtualBox: ~/Downloads
cesar@cesar-VirtualBox:~/Downloads$ ls
linux-source-4.15.0  linux-source-4.15.0.tar.bz2
cesar@cesar-VirtualBox:~/Downloads$
```

Figura 4 - Print mostrando arquivo descompactado

Antes de prosseguir será necessário instalar algumas bibliotecas para que a compilação possa ser feita de maneira correta. Abaixo estão as bibliotecas necessárias e seus respectivos comandos para instalação.

- libncurses5: `'sudo apt-get install libncurses5-dev'`
- build-essential: `'sudo apt-get install build-essential'`
- libssl: `'apt install libssl-dev -y'`

Com as bibliotecas instaladas podemos dar continuidade ao processo de compilação.

Próximo passo é virar root da máquina. Para isso digite o comando `'sudo su'`. Virando root da máquina os passos finais da compilação poderão ser iniciados. Primeiro comando a se digitar é o `'make menuconfig'`. Com esse comando uma tela se abrirá com as configurações do linux. Você poderá navegar nesse menu desativando módulos que são desnecessários em sua máquina, diminuindo assim o tempo de compilação total.

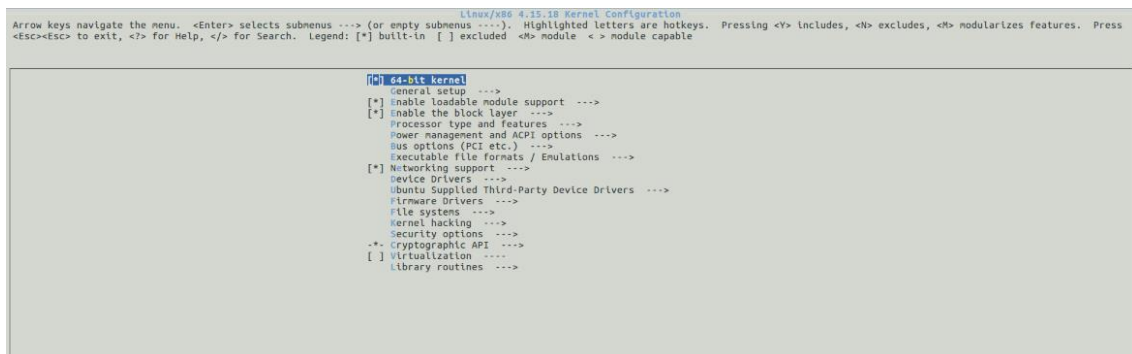
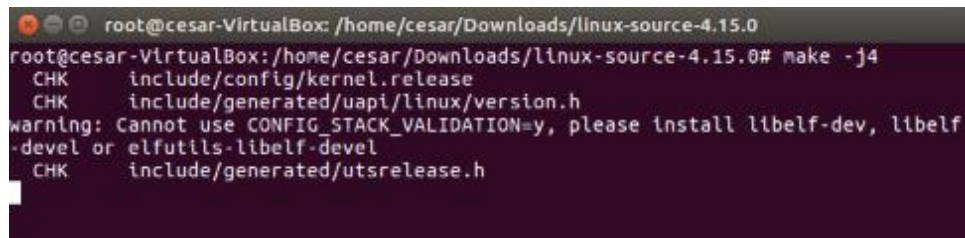


Figura 5 - Print da tela mostrando o menu de configuração do kernel

No topo do menu há pequenas instruções, indicando como navegar pelo menu, como ativar/desativar módulos, etc. O importante nessa fase é analisar o que é e o que não é essencial para o kernel. A função da grande maioria dos módulos é desconhecida, portanto há o risco de algo fundamental ser desativado. Para que esse erro não compromettesse o desenvolvimento do experimento, foram sendo criadas cópias da máquina virtual, conforme obtivemos versões de kernel que funcionassem adequadamente.

Após a desativação dos módulos, é necessário compilar o kernel de fato com suas novas configurações. Para que isso ocorra digite o comando `'make'` no

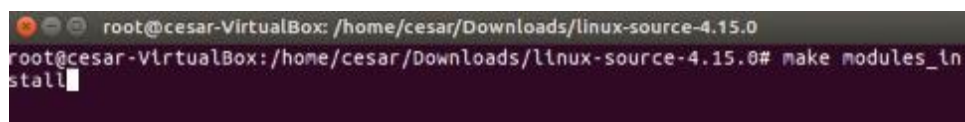
terminal. Caso sua máquina virtual esteja habilitada para rodar com mais de uma CPU, você pode usar o comando `'make -jN'`, sendo que 'N' é o número de CPUs que você tem disponível para a máquina virtual. Esse comando, como citado, é responsável pela compilação, criando um binário do kernel.



```
root@cesar-VirtualBox: /home/cesar/Downloads/linux-source-4.15.0
root@cesar-VirtualBox:/home/cesar/Downloads/linux-source-4.15.0# make -j4
CHK include/config/kernel.release
CHK include/generated/uapi/linux/version.h
warning: Cannot use CONFIG_STACK_VALIDATION=y, please install libelf-dev, libelf
-devel or elfutils-libelf-devel
CHK include/generated/utsrelease.h
```

Figura 6 - Print mostrando o comando 'make'

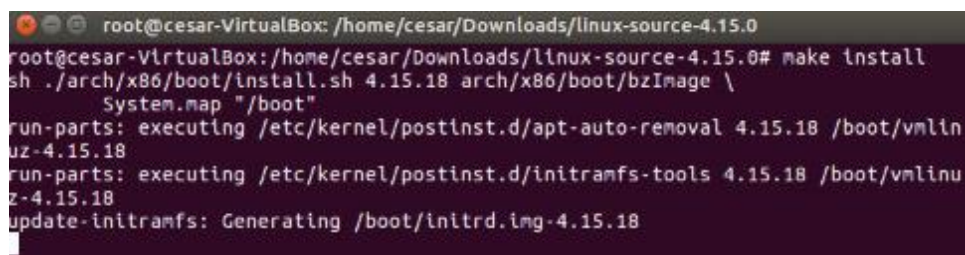
Após se compilar o kernel é necessário fazer uma instalação dinâmica dos módulos do kernel. Para isso é usado o comando `'make modules_install'`.



```
root@cesar-VirtualBox: /home/cesar/Downloads/linux-source-4.15.0
root@cesar-VirtualBox:/home/cesar/Downloads/linux-source-4.15.0# make modules_in
stall
```

Figura 7 - Print mostrando o comando 'make modules_install'

Após este comando basta apenas instalar o kernel em sua máquina com o comando `'make install'`.



```
root@cesar-VirtualBox: /home/cesar/Downloads/linux-source-4.15.0
root@cesar-VirtualBox:/home/cesar/Downloads/linux-source-4.15.0# make install
sh ./arch/x86/boot/install.sh 4.15.18 arch/x86/boot/bzImage \
System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 4.15.18 /boot/vmlin
uz-4.15.18
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 4.15.18 /boot/vmlinu
z-4.15.18
update-initramfs: Generating /boot/initrd.img-4.15.18
```

Figura 8 - Print mostrando o comando 'make install'

Após a instalação ser concluída reinicie sua máquina com o comando `'reboot'`.

Possíveis erros durante a compilação do kernel

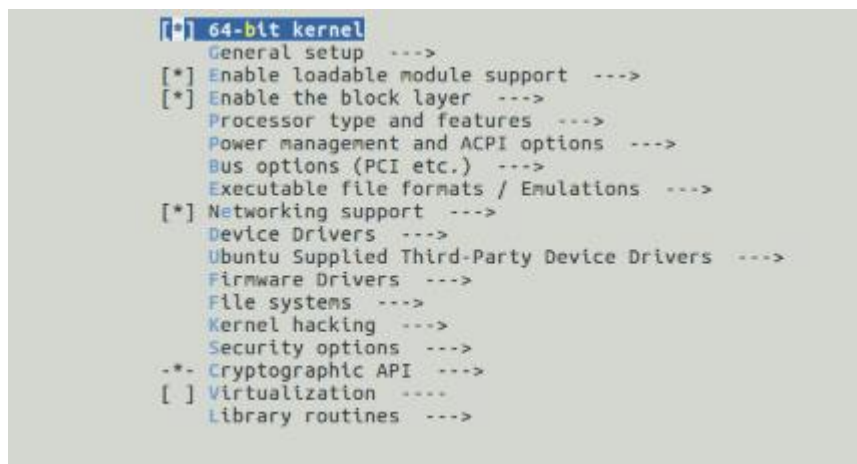
- *Makefile: 1086: recipe for target './drivers' failed:* quando a compilação do kernel é finalizada é possível se deparar com esse erro, no qual alguns drivers retornavam erro. O próprio ubuntu nos dá a solução, dizendo que é necessário instalar a lib bison. Para instalar a biblioteca, usa-se o comando `'sudo apt-get install bison flex'`. Anteriormente citamos algumas bibliotecas que precisam ser instaladas para a compilação do kernel. Há casos em que esta biblioteca não precisa ser instalada, portanto não está citada anteriormente.
- *Make: *** [menuconfig] Error 2:* Esse erro só irá ocorrer caso não se instale a biblioteca libssl.
- *Make: *** No rule to make target 'menuconfig'. Stop:* Esse erro ocorre ao tentar abrir o menu para ter acesso ao kernel para modificá-lo. Ocorre

caso você não esteja como root da máquina ou esteja tentando rodar o comando *'make menuconfig'* na pasta errada.

- *Kernel Panic - Not syncing: out of memory and no killable process*: Esse erro ocorre na reinicialização (reboot) da máquina após o kernel ser compilado. Para este erro de kernel panic, há a necessidade de adicionar mais memória à sua máquina virtual.

Alguns dos módulos desinstalados

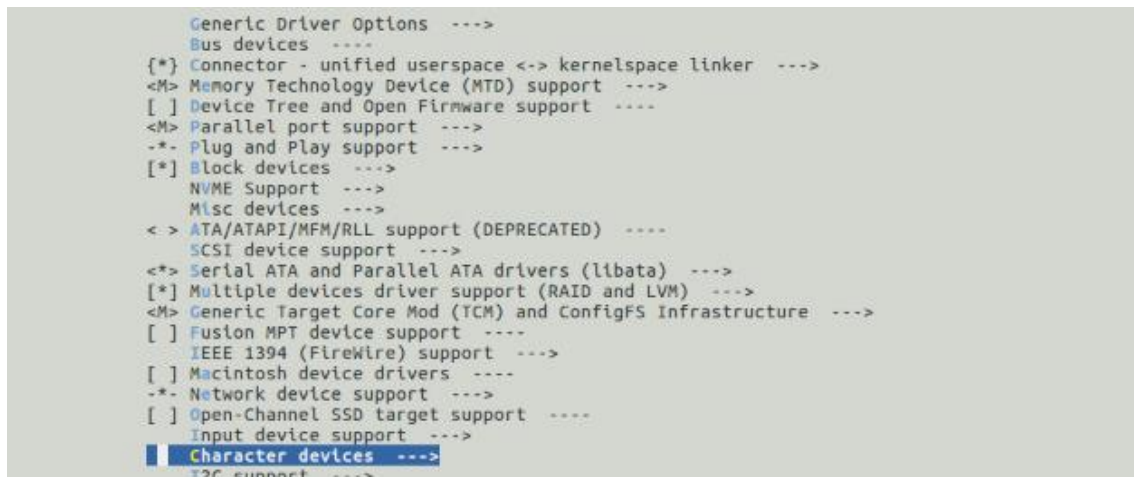
Como citado anteriormente é necessário desativar alguns módulos do kernel para que este compile mais rapidamente. Abaixo estão alguns prints mostrando apenas uma parte dos módulos desativados.

A screenshot of the 'make menuconfig' interface for a 64-bit kernel. The '64-bit kernel' option is selected. Under the 'Networking support' section, which is currently disabled, the 'Device Drivers' sub-section is highlighted. Within 'Device Drivers', the 'Virtualization' option is shown as disabled, indicated by a space in front of the bracketed asterisk. Other options like 'Cryptographic API' and 'Library routines' are also visible in the list.

```
[*] 64-bit kernel
    General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
    Processor type and features --->
    Power management and ACPI options --->
    Bus options (PCI etc.) --->
    Executable file formats / Emulations --->
[*] Networking support --->
    Device Drivers --->
    Ubuntu Supplied Third-Party Device Drivers --->
    Firmware Drivers --->
    File systems --->
    Kernel hacking --->
    Security options --->
    *- Cryptographic API --->
[ ] Virtualization ---
    Library routines --->
```

Figura 9 - Print da tela de configuração do kernel

Nesta imagem podemos ver que o driver de virtualização foi desativado, uma vez que já estamos em uma máquina virtual.

A screenshot of the 'make menuconfig' interface showing various device drivers. The 'Character devices' section is highlighted. Within this section, the 'I2C support' option is shown as disabled, indicated by a space in front of the bracketed asterisk. Other options like 'Parallel port support' and 'Plug and Play support' are also visible.

```
Generic Driver Options --->
Bus devices ---
[*] Connector - unified userspace <-> kernelspace linker --->
<M> Memory Technology Device (MTD) support --->
[ ] Device Tree and Open Firmware support ---
<M> Parallel port support --->
*- Plug and Play support --->
[*] Block devices --->
    NVME Support --->
    Misc devices --->
< > ATA/ATAPI/MFM/RLL support (DEPRECATED) ---
SCSI device support --->
<*> Serial ATA and Parallel ATA drivers (libata) --->
[*] Multiple devices driver support (RAID and LVM) --->
<M> Generic Target Core Mod (TCM) and ConfigFS Infrastructure --->
[ ] Fusion MPT device support ---
    IEEE 1394 (FireWire) support --->
[ ] Macintosh device drivers ---
*- Network device support --->
[ ] Open-Channel SSD target support ---
Input device support --->
[*] Character devices --->
    I2C support ---
```

Figura 10 - Print da tela de configuração do kernel


```

[ ] SPI support ----
<M> SPMI support ----
<M> HSI support --->
{M} PPS support --->
    PTP clock support --->
-* Pin controllers --->
-* GPIO Support --->
{M} Dallas's 1-wire support --->
[*] Adaptive Voltage Scaling class support --->
[*] Board level reset or power off --->
-* Power supply class support --->
[*] Hardware Monitoring support --->
-* Generic Thermal sysfs driver --->
[*] Watchdog Timer Support --->
    Sonics Silicon Backplane --->
<M> Broadcom specific AMBA --->
Multifunction device drivers --->
-* Voltage and Current Regulator Support --->
< > Remote Controller support ----
< > Multimedia support ----
    Graphics support --->
< > Sound card support ----
    HID support --->
[*] USB support --->

```

Figura 11 - Print da tela de configuração do kernel

As duas imagens anteriores mostram um pouco dos device drivers do kernel. Aqui estão a maioria dos drivers que podem ser desativados, há diversas opções, desde de drivers de internet até suporte USB.

```

< > Analog Devices Digital Potentiometers
< > Dummy IRQ handler
< > Device driver for IBM RSA service processor
< > Sensable PHANTOM (PCI)
< > SGI IOC4 Base IO support
< > TI Flash Media interface support
< > Integrated Circuits ICS9325401
< > Enclosure Services
< > Channel interface driver for the HP iLO processor
< > Medfield Avago APDS9802 ALS Sensor module
< > Intersil ISL29003 ambient light sensor
< > Intersil ISL29020 ambient light sensor
< > Taos TSL2550 ambient light sensor
< > BH1770GLC / SFH7770 combined ALS - Proximity sensor
< > APDS990X combined als and proximity sensors
< > Honeywell HMC6352 compass
< > Dallas DS1602 Total Elapsed Time Recorder with Alarm
< > FSA9480 USB Switch
[ ] Generic on-chip SRAM driver
< > PCI Endpoint Test driver
< > Silicon Labs C2 port support ----
    EEPROM support --->
< > ENE CB710/720 Flash memory card reader support
    Texas Instruments shared transport line discipline --->
< > STMicroelectronics LIS3LV02Dx three-axis digital accelerometer (I2C)
< > Altera FPGA firmware download module
< > Intel Management Engine Interface
< > ME Enabled Intel Chipsets
< > Intel Trusted Execution Environment with ME Interface
< > VMware VMCI Driver
    Intel MIC & related support --->
< > GenWQE PCIe Accelerator ----
< > Line Echo Cancellation support
< > Realtek PCI-E card reader
< > Realtek USB card reader

```

Figura 12 - Print da tela de configuração do kernel


```

<M> A4 tech nice
< > Accutouch touch device
< > ACRUX game controller support
< > Apple {l,Power,Mac}Books
< > Apple infrared receiver
< > Asus
< > Aureal
< > Belkin Flip KVM and Wireless keyboard
< > Betop Production Inc. force feedback support
< > Cherry Cymotion keyboard
< > Chicony devices
< > Corsair devices
< > CMedia CM6533 HID audio jack controls
<M> Silicon Labs CP2112 HID USB-to-SMBus Bridge support
<M> Cypress mouse and barcode readers
< > DragonRise Inc. game controller
< > EMS Production Inc. force feedback support
< > ELECOM HID devices
< > ELO USB 4000/4500 touchscreen
< > Ezkey BTC 8193 keyboard
< > Gembird Joypad
< > Google Fiber TV Box remote control support
< > Holtek HID devices
< > MSI GT68XR LED support
<M> Keytouch HID devices
<M> KYE/Genius devices
<M> UC-Logic
< > Waltop

```

Figura 13 - Print da tela de configuração do kernel

```

--- LED Support
[*] LED Class Support
< > LED Flash Class Support
[ ] LED Class brightness_hw_changed attribute support
*** LED drivers ***
< > LED Support for Marvell 88PM860x PMIC
< > Front panel LED support for PC Engines APU/APU2 boards
< > LCD Backlight driver for LM3530
< > LED support for LM3533
< > LED support for LM3642 Chip
< > LED Support for Mediatek MT6323 PMIC
< > LED driver for PCA9532 dimmer
< > LED Support for GPIO connected LEDs
< > LED Support for N.S. LP3944 (Fun Light) I2C chip
< > LED Support for TI LP3952 2 channel LED driver
< > LED Support for N.S. LP5521 LED driver chip
< > LED Support for TI/National LP5523/55231 LED driver chip
< > LED Support for TI LP5562 LED driver chip
< > LED Support for TI LP8501 LED driver chip
< > LED support for the TI LP8788 PMIC
< > LED support for the TI LP8860 4 channel LED driver
< > Mail LED on Clevo notebook
< > LED Support for PCA955x I2C chips
< > LED support for PCA963x I2C chip
< > LED Support for DA9030/DA9034 PMIC

```

Figura 14 - Print da tela de configuração do kernel

As imagens anteriores mostram algumas opções dentro do menu de device drivers. Como citado anteriormente a tela de configuração do kernel é muito vasta, com diversas opções de módulos para serem desativados. O importante é ir compilando aos poucos e verificando os resultados, quanto mais prática compilando o kernel, mais fácil será para você modificá-lo.

Conclusão

A maior parte dos módulos de um kernel são desconhecidos pela maioria das pessoas, até mesmo para aquelas que trabalham no setor de computação. Portanto, para a compilação de um kernel enxuto era necessário se arriscar e desativar módulos que possivelmente são essenciais para o kernel. Esse processo leva a diversos erros, que são de extrema importância para o entendimento do que poderia e o que não poderia ser feito dentro de um kernel. Com esses e outros erros, foi possível compreender todos os passos para compilação de um kernel de forma clara.