**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS**

**CENTRO DE CIÊNCIAS EXATAS, AMBIENTAIS E DE TECNOLOGIA**

**CHRISTOPHER OLIVEIRA RA:18726430**

**GIULIANO SANFINS RA:17142837**

**MATHEUS MORETTI RA:18082974**

**MURILO ARAUJO RA:17747775**

**VICTOR REIS RA:18726471**

**SISTEMAS OPERACIONAIS B - EXPERIMENTO 1 -COMPILAÇÃO DE KERNEL (LINUX)**

**CAMPINAS**

**2020**

**SUMÁRIO**

[1. INTRODUÇÃO 3](#_Toc48825413)

[2. DISCUSSÃO 3](#_Toc48825414)

[2.1 Tempos de Execução: 9](#_Toc48825415)

[2.2 Dificuldades: 9](#_Toc48825416)

[3. CONCLUSÃO 10](#_Toc48825417)

# **1. INTRODUÇÃO**

O experimento tem como objetivo principal a compilação de um Kernel Linux, permitindo através da mesma, que nós tenhamos um maior conhecimento e se adaptar a manipular e alterar recursos do Kernel, pois do decorrer do semestre vamos estar trabalhando e elaborando atividades mais complexas com o Kernel. Além de realizar a compilação, é esperado também com essa atividade que sejamos capazes de realizar a instalação e testes com um novo Kernel Linux.

E também estar entendendo todo o funcionamento de cada passo a passo do processo de compilação do kernel, conhecendo também, o tempo necessário para cada passo e os comandos e ferramentas utilizadas.

# **2. DISCUSSÃO**

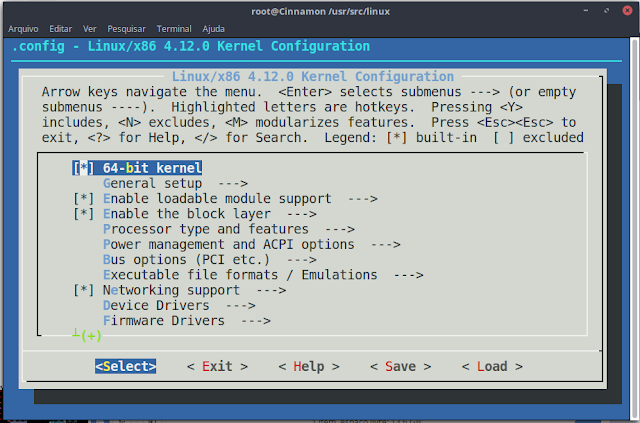
O primeiro passo foi a instalação de uma máquina virtual com o objetivo de simular um computador. O uso de máquinas virtuais traz como vantagem o menor uso de servidores e, com isso, uma economia em compra de equipamentos, em sua manutenção, em energia e em refrigeração do ambiente. Também reduz custos relacionados à instalação, atualização e manutenção de servidores. O segundo passo foi o download do kernel Ubuntu 4.1.15 usando os comandos:

apt-cache search linux-source  
apt-get versão

Depois da descompactação usamos o comando:

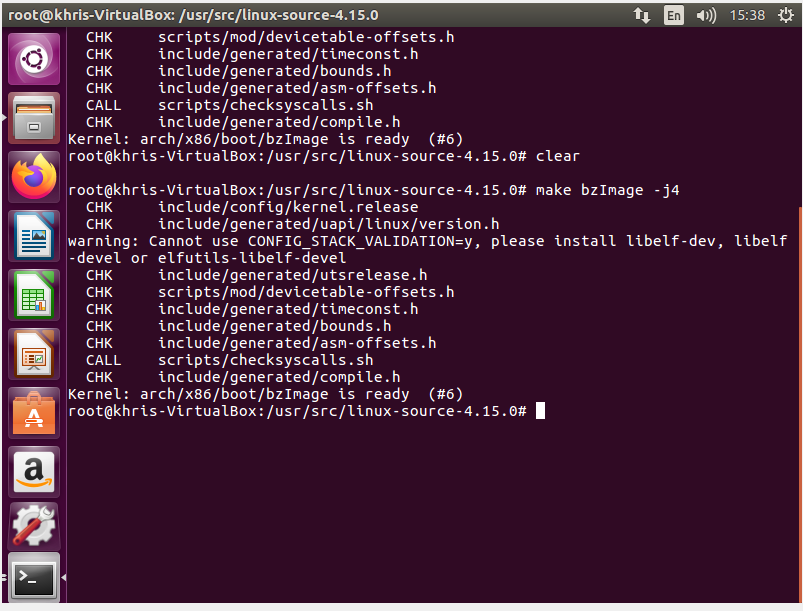
make menuconfig

O comando “*make menuconfig*” é uma ferramenta para que se possa compilar o código fonte do Linux. O lado positivo desse comando é que existe uma interface gráfica e intuitiva mostrando todas as configurações do Kernel(núcleo), como por exemplo, drivers de rede, drivers de vídeo, USB, WI-FI e etc.  
Exemplo da interface mostrada no *menuconfig*:



Um dos maiores objetivos desse experimento é modificar o Kernel para deixá-lo o mais enxuto possível para que a compilação seja rápida para o hardware desejado, sendo assim, necessário remover muitos drivers e configurações presentes no código fonte original do Linux. Essas configurações são salvas em um arquivo chamado .config” (Anexado com o relatório).

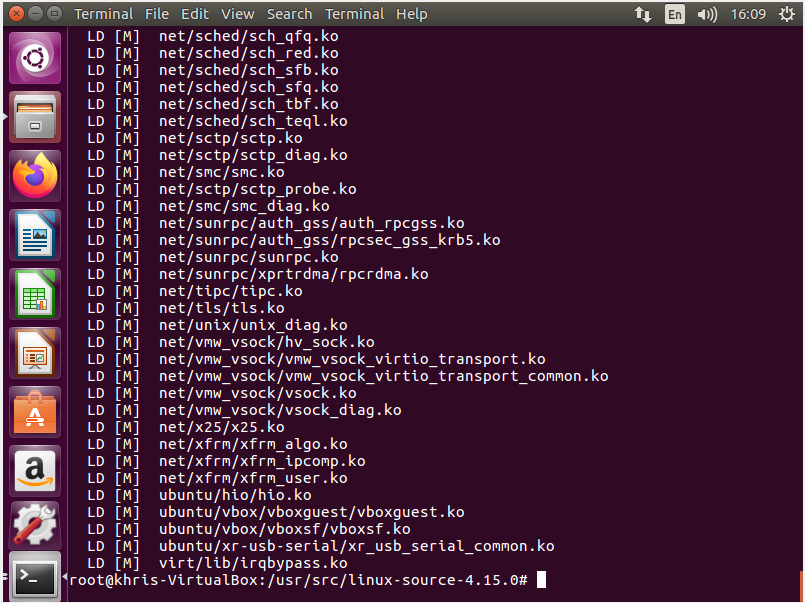
Depois de selecionar as configurações desejadas é necessário compilar o Kernel e deixar um arquivo no diretório *arch/x86\_62/boot* chamado “*bzImage*”, que é o kernel comprimido.



make bzImage

Após comprimir o kernel é necessário executar o comando “make modules\_install” que garantirá a compilação dos módulos e instalar os módulos (binários) na pasta do kernel.

make modules\_install



Agora que o Kernel já está compilado, os arquivos já podem ser copiados para seus destinos finais. O comando “*make install*” copiará o programa construído, *libraries* e documentação para seus destinos corretos.

make install

Depois que os módulos foram instalados precisamos “montar” o root *filesystem*, e para isso foi usado o comando *initramfs*.

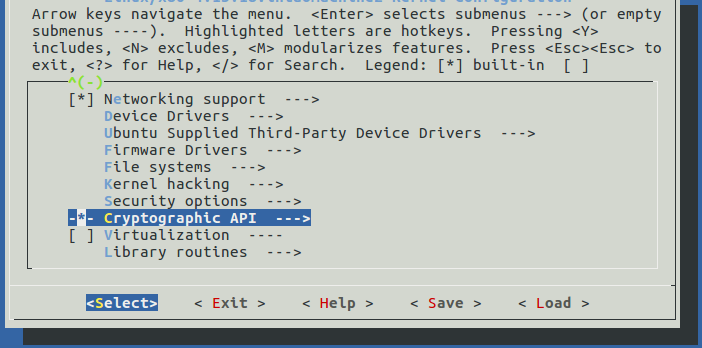
mkinitramfs -o initrd.img-4.12(use o tab para completar) (nome do kernel)

Com tudo instalado e montado só é preciso dar um upgrade no grub.

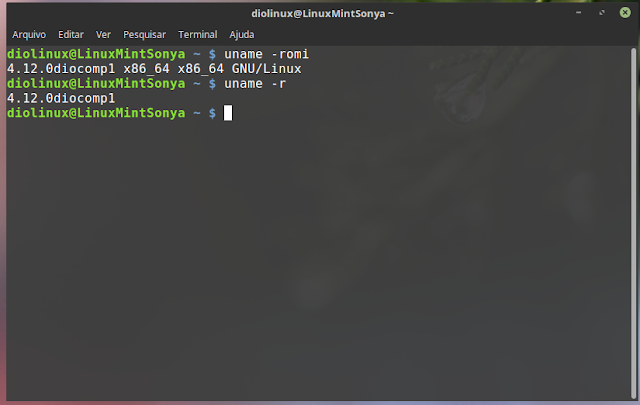
upgrade-grub

Feito tudo isso, já temos a imagem do kernel e do initrd encontradas com a nossa compilação nós já temos tudo pronto para começar a utilizar e testar o nosso kernel compilado. Após o reboot pode ser feito a verificação se o Kernel customizado foi realmente instalado com o comando:

uname -r

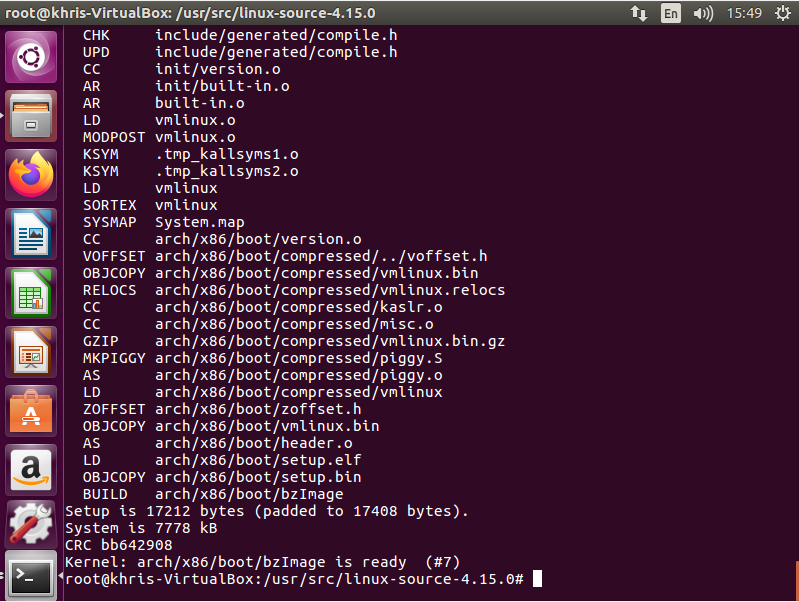


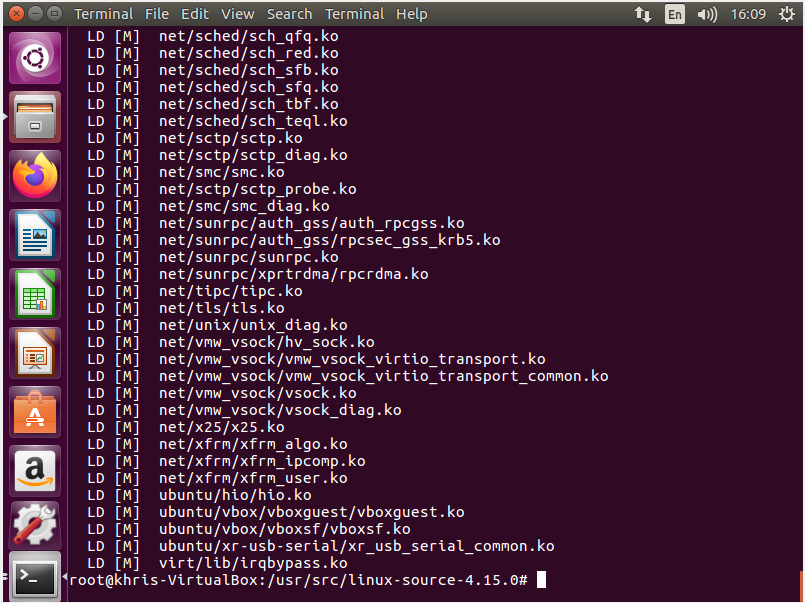
E o resultado deve ser algo parecido (se você mudou o nome da versão que fica armazenado no arquivo Makefile);



Interface do MenuConfig.

Retorno após criação da Imagem.



Retorno do Comando Make Modules.

Também foi realizado *overclock* na máquina através das configurações da *bios* e utilizado o comando abaixo para escolher 4 (quatro) núcleos para a realizar a compilação do kernel da máquina virtual.

-JN (Número de núcleo)

## **2.1 Tempos de Execução:**

Computador 1.

*Config*:

Processador: i5-7600k (4.9Ghz) (Com OC)

Memória RAM: 20Gb.

Execuções e seus tempos:

Imagem do Kernel: 6:20 min.

Compilação do Kernel: 12:25 min.

Tempo total: 18:45 min.

## **2.2 Dificuldades:**

* Superaquecimento do processador, chegando até 95ºC.
* Processamento do computador.

# **3. CONCLUSÃO**

Com esse experimento pudemos conhecer mais sobre o *linux* no que se diz respeito ao kernel, como procurar kernels disponíveis, baixá-lo e instalá-lo, do mesmo modo aprendemos a como verificar e alterar as configurações presentes no *kernel*, e através disso pudemos concluir um dos principais objetivos do experimento, que foi deixar todo processo de compilação de modules, e instalação do kernel muito mais rápido. Assim de forma geral inferimos que fora alcançado todos os objetivos propostos para esse experimento, tais como, aprender mais sobre o sistema *linux* voltado a âmbito de kernel, aprender os processos envolvidos desde a busca de um kernel até sua configuração e instalação para uso, e a como distinguir quais opções presentes nas configurações são realmente necessárias para que possamos cumprir nosso futuros objetivos de forma que exigirá menos tempo.