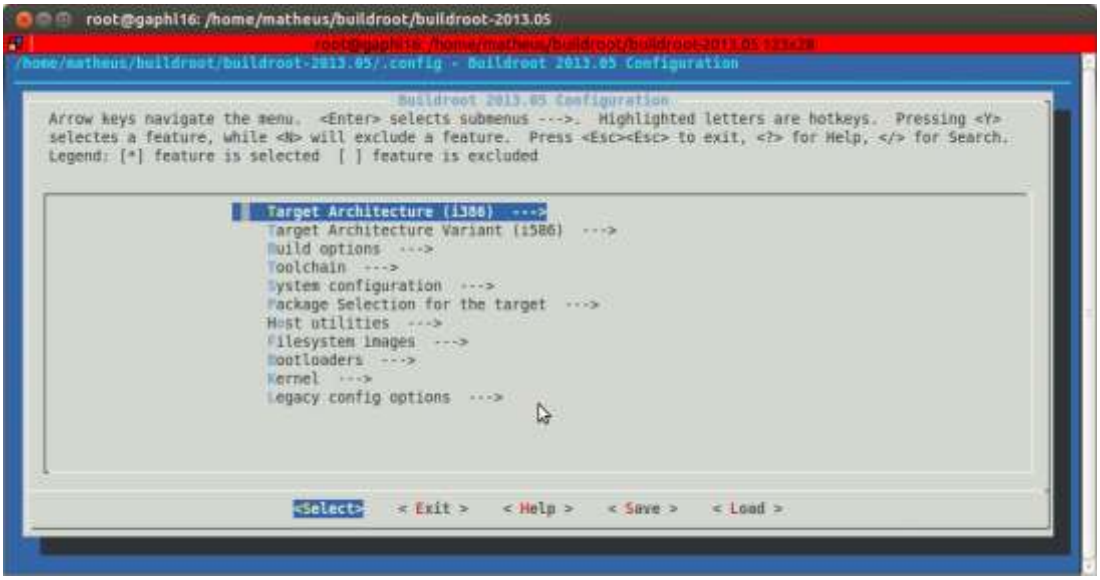
Tutorial BuildRoot

Programação de Periféricos – CC – FACIN – PUCRS

Configuração do Ambiente de Trabalho:

1. Criar um diretório de trabalho:
   Ex: *mkdir ~/buildroot*
2. Ir para o diretório de trabalho:
   Ex: *cd ~/buildroot*
3. Baixar o BuildRoot do fornecedor:
   Ex: *wget http://buildroot.uclibc.org/downloads/buildroot-2013.05.tar.gz*
4. Extrair:
   Ex: *tar xvzf buildroot-2013.05.tar.gz*
5. Ir para o diretório:
   Ex: *cd buildroot-2013.05*
6. Executar o BuildRoot
   Ex: *make menuconfig*

Se tudo deu certo, deverá estar disponível um diretório de trabalho contendo o BuildRoot e a seguinte tela deve aparecer:

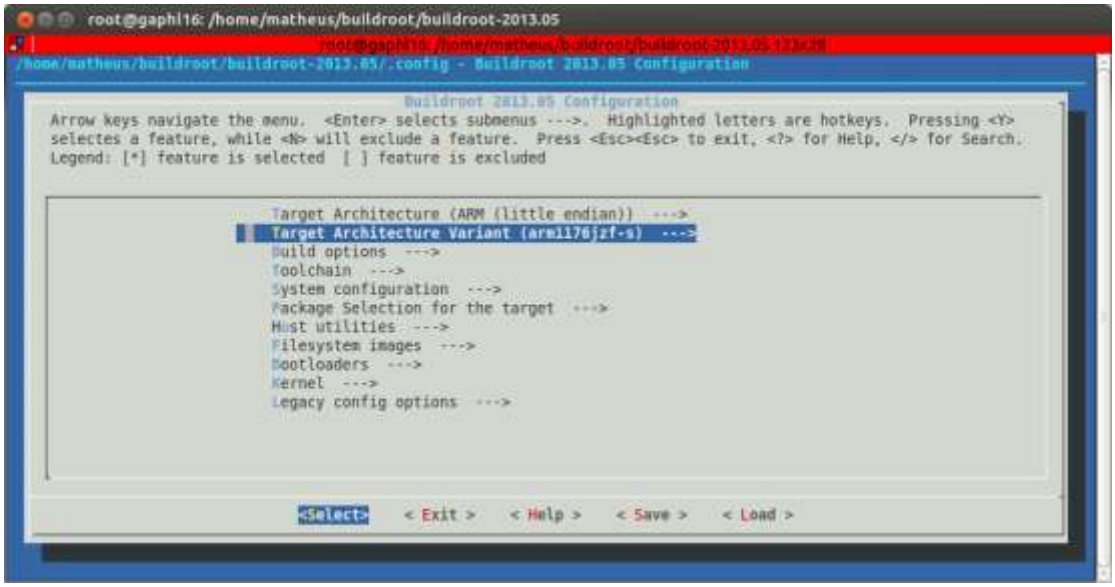


Meu Primeiro Linux embarcado:

Configurar a arquiteutura alvo de acordo com a Raspberry Pi:

Target Architecture: ARM (little endian)
Target Architecture Variant: arm1176jzf-s

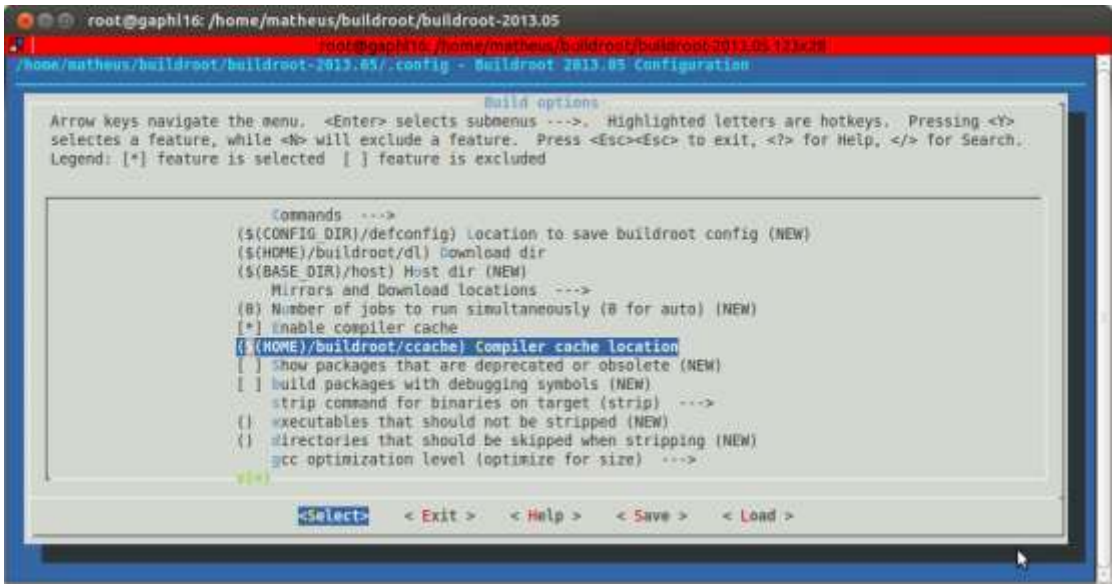
O resultado deve ser equivalente ao da figura abaixo:



Configurar opções do BuildRoot:

Selecionar a opção ***Build Options*** do menu.
Download dir: $(HOME)/buildroot/dl
Enable compiler cache: YES
Compiler cache location: $(HOME)/buildroot/ccache

O resultado deve ser equivalente ao da figura abaixo:

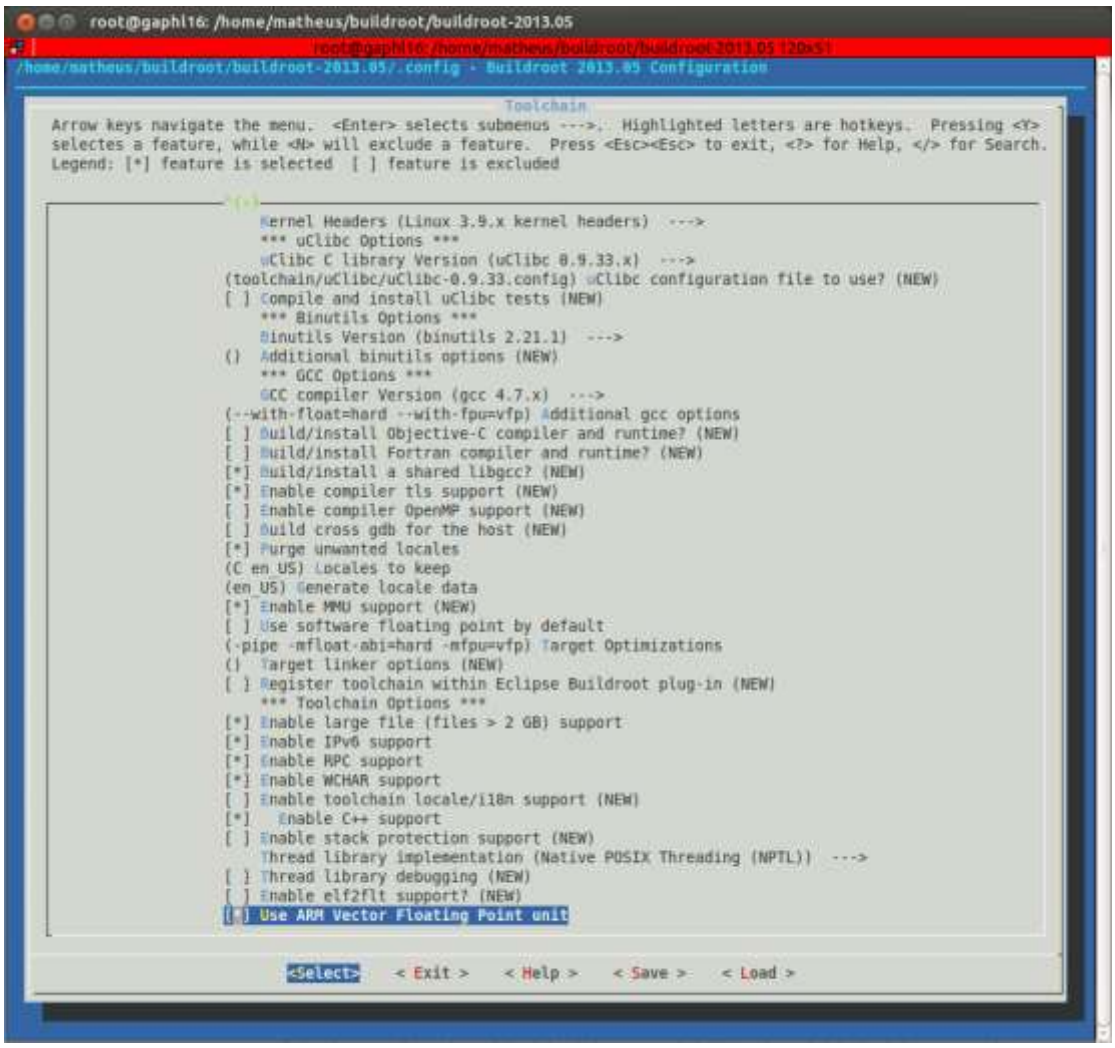


Configurar opções do Toolchain:

Selecionar a opção ***Toolchain*** do menu.
Kernel Headers: Linux 3.9.x kernel headers
GCC compiler Version: GCC 4.8.x
Additional gcc options: --with-float=hard --with-fpu=vfp
Configurações do GCC:
Purge unwanted locales YES

Locales to keep C en_US
Generate locale data en_US
Use software floating point by default: NO
Target Optimizations: -pipe -mfloat-abi=hard -mfpu=vfp
Use ARM Vector Floating Point unit: YES
Enable large file (files > 2 GB) support: YES
Enable IPv6 support: YES
Enable RPC support: YES
Enable WCHAR support: YES
Enable C++ support: YES

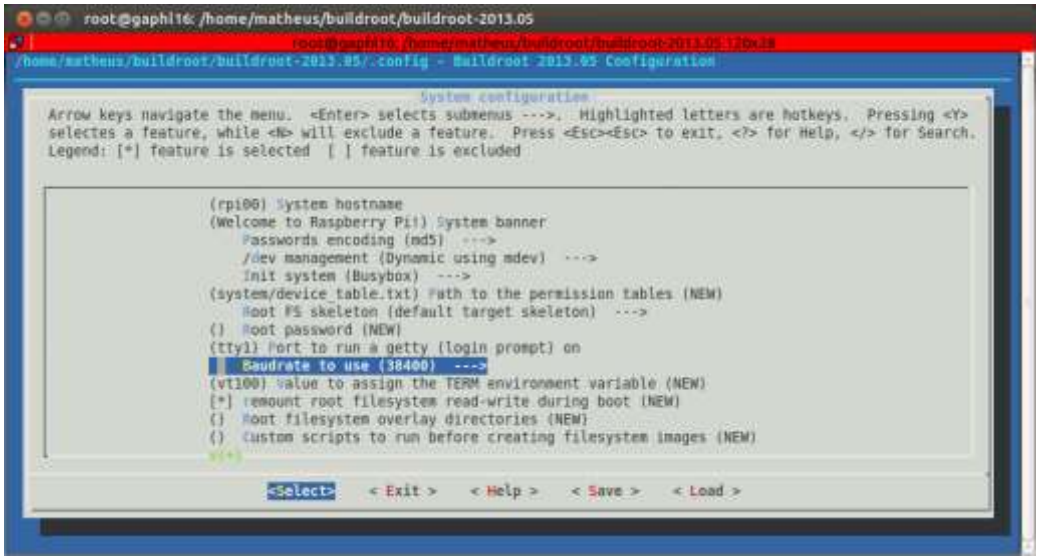
O resultado deve ser equivalente ao da figura abaixo:



Configurar opções de Sistema embarcado:

Selecionar a opção ***System Configuration*** do menu.
System hostname rpi00 ***(NUMERO DO GRUPO)***
System banner Welcome to Raspberry Pi!
/dev management Dynamic using mdev
Port to run a getty (login prompt) on tty1
Baudrate to use 38400

O resultado deve ser equivalente ao da figura abaixo:



Configurar opções de Pacotes a serem disponibilizados na distribuição criada:

Selecionar a opção ***Packet Selection ...*** do menu.

Busybox deve estar habilitado. Esse pacote provê um conjunto completo de ferramentas básicas para utilizar o linux.

Adicionar um serviço para SSH:

Networking applications → dropbear

Adicionar também um serviço para NFS:

Networking applications → portmap

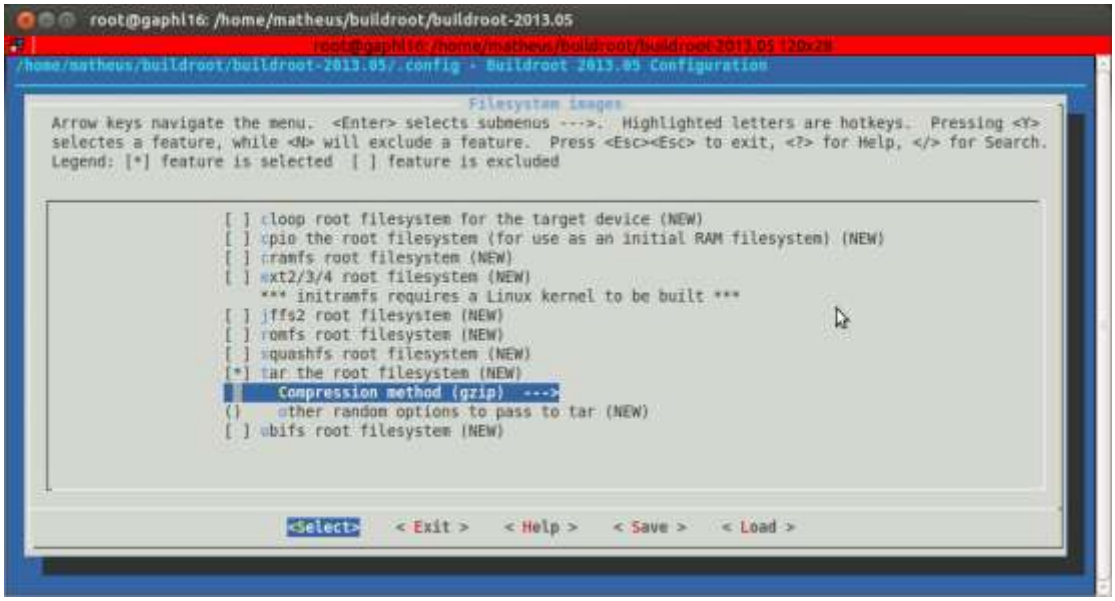
O resultado deve ser equivalente ao da figura abaixo:

Configurar opções de Imagens:

Selecionar a opção ***Filesystem Images*** do menu.

Compression method: gzip

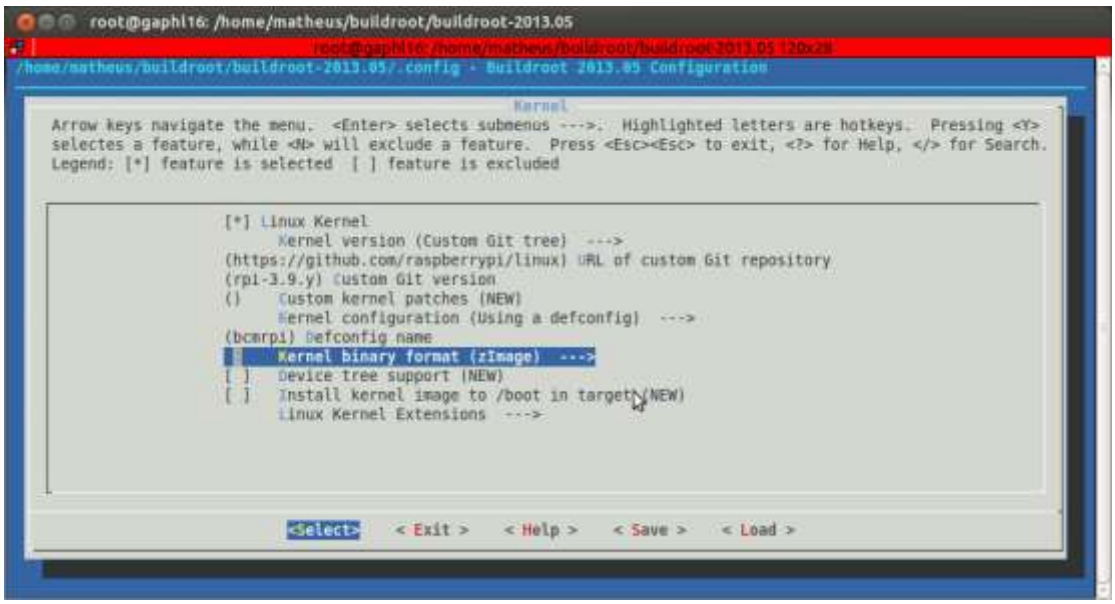
O resultado deve ser equivalente ao da figura abaixo:



Configurar opções de Kernel:

Selecionar a opção ***Kernel*** do menu.

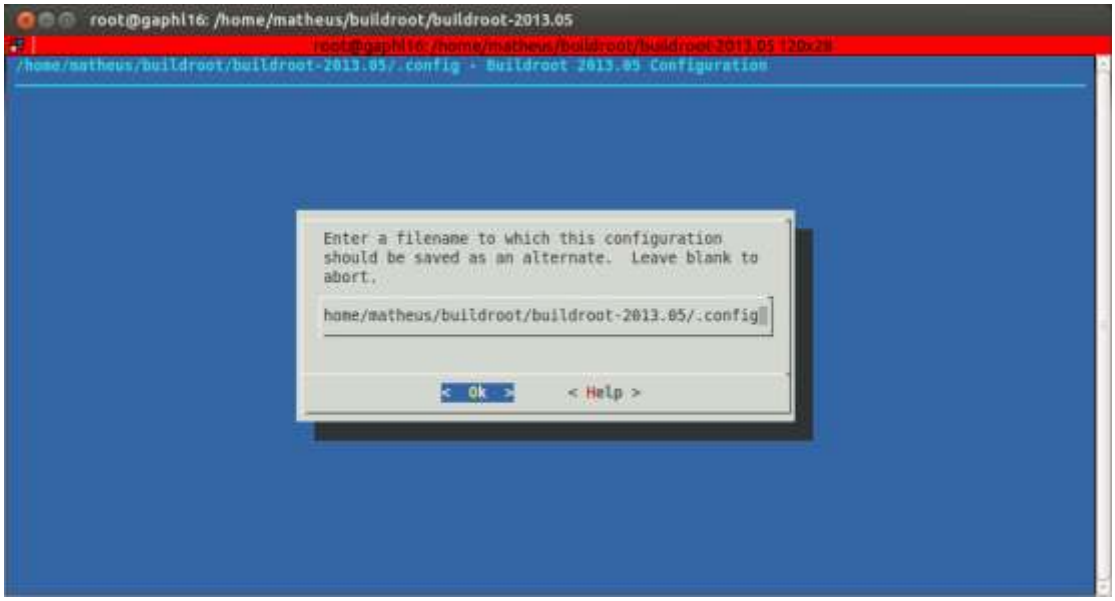
Linux Kernel	YES
Kernel version	Custom Git tree
URL of custom Git repository	https://github.com/raspberrypi/linux
Custom Git version	rpi-3.9.y
Kernel configuration	Using a defconfig
Defconfig name	bcmrpi
Kernel binary format	zImage



O resultado deve ser equivalente ao da figura abaixo:

Gerar o Linux a ser embarcado na plataforma:

Salvar as configurações em ***Save*** no menu.



Sair do menu de configurações em ***EXIT***.

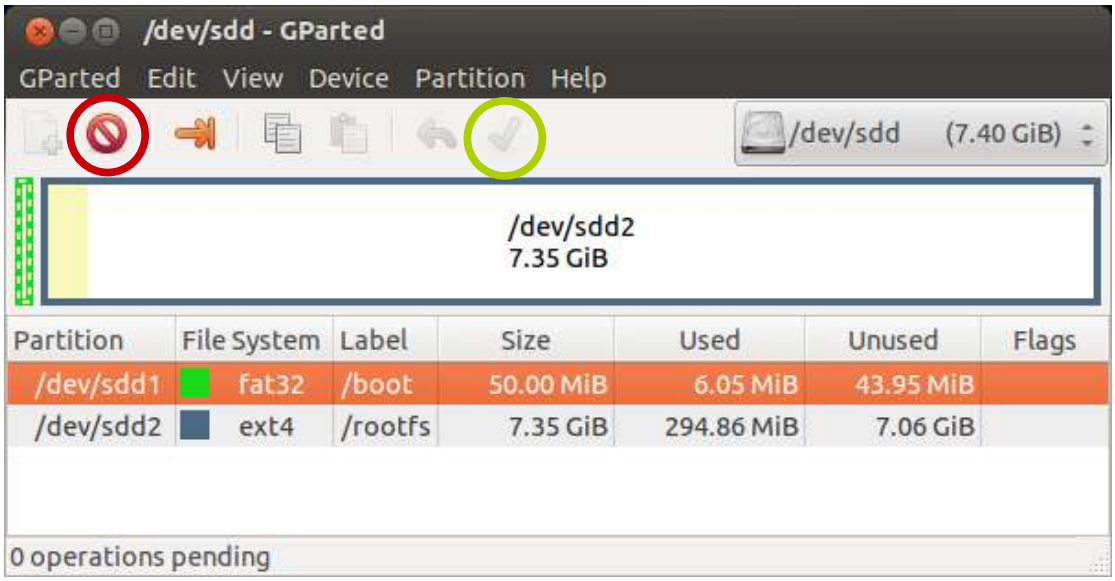
Gerar o sistema, digitando o seguinte comando no terminal:
make all

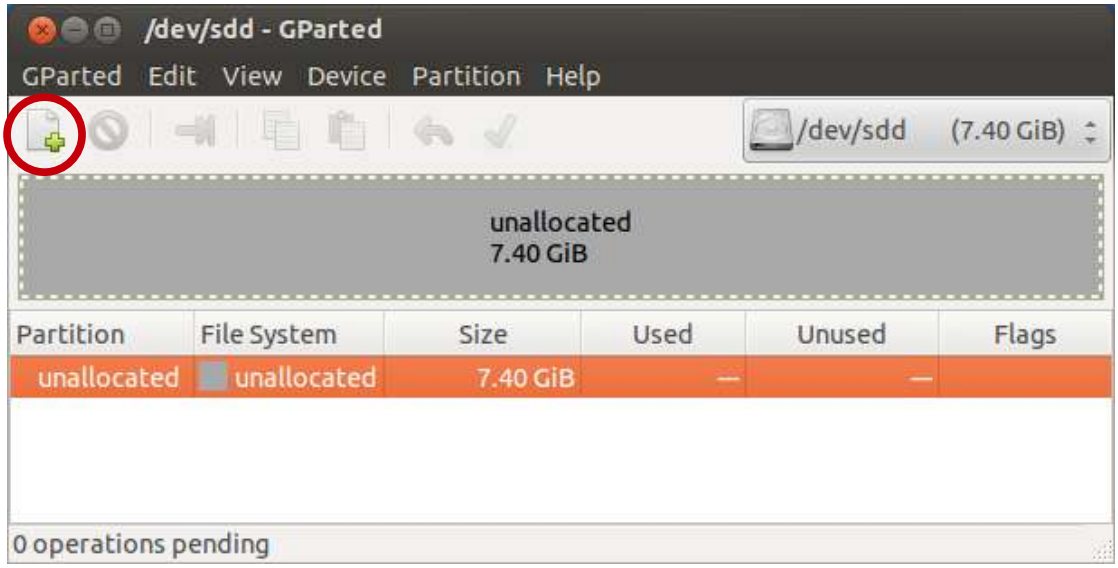
Essa etapa pode demorar vários minutos.
Talvez seja necessário instalar o g++ e o git:
apt-get install g++ git

Preparação do SD Card:

Enquanto é feita a compilação, pode-se adiantar a tarefa de gerar as partições necessárias no cartão SD. Para tanto, utilizar a ferramenta gparted.

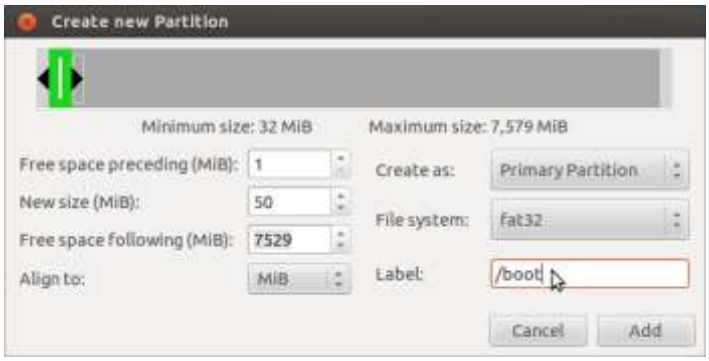
Primeiramente, selecionar o cartão SD, que deve ter ~7,4GB, e deletar qualquer partição que exista no cartão, isso pode ser feito com o botão destacado na figura abaixo e aplicar as modificações:



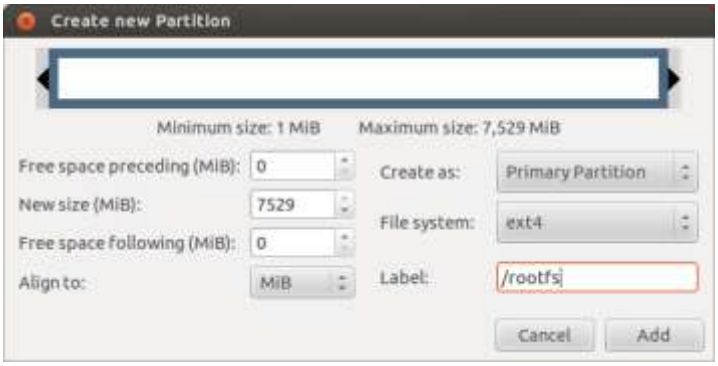


Crie duas novas partições clicando no botão destacado acima.

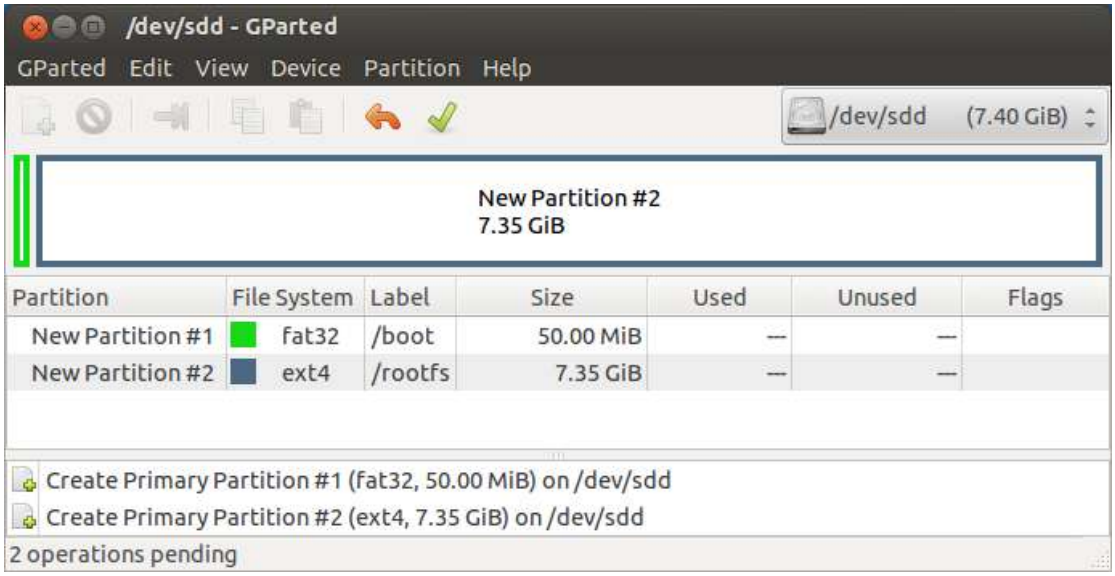
A primeira partição deverá ser chamada */boot* e deverá ser configurada como *fat32* de *50MB*. Além disso, essa partição deve ser a primeira partição do cartão, conforme demonstra a figura abaixo:



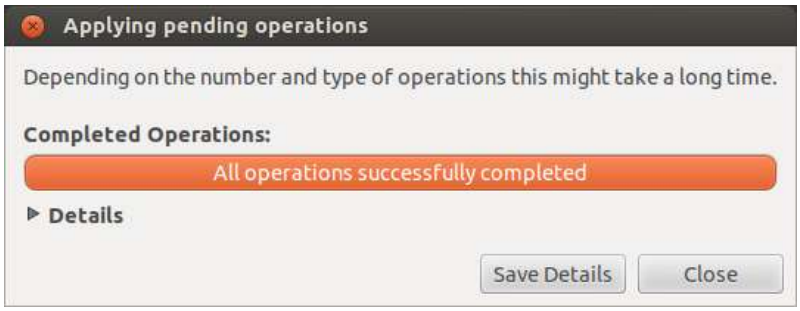
Em seguida, crie outra partição para conter o sistema de arquivos. Essa partição deverá ser chamada */rootfs* e deverá ser configurada como *ext4*. Seu tamanho deverá ser o restante do cartão, conforme demonstra a figura abaixo:



O resultado deve ser equivalente ao da figura abaixo:



Aplicar as modificações e fechar a janela que segue:



Criar pontos de montagem e montar o cartão:
mkdir /media/rootfs
mount /dev/sdd2 /media/rootfs
mkdir /media/boot
mount /dev/sdd1 /media/boot

Cuidado para montar o dispositivo correto. No caso desse exemplo, o cartão foi montado em /dev/sdd e as partições /boot e /rootfs foram montadas em /dev/sdd1 e /dev/sdd2, respectivamente. Os dispositivos disponíveis podem ser visualizados utilizando o seguinte comando:
df -h

Notar que até a conclusão do BuildRoot, as seguintes etapas serão executadas:
1. Um toolchain será gerado para a máquina onde o BuildRoot está sendo executado;
2. Um toolchain será gerado para compilar descrições e gerar códigos para a arquitetura alvo (ARM). Ou seja, será gerado um ambiente que permite *cross-compilar* códigos para o processador da Raspberry Pi;
3. Baixar, configurar e compilar todos pacotes selecionados para o Linux a ser gerado, utilizando o *cross-compiler* e a biblioteca µClib;
4. Instalar os pacotes;
5. Criar uma imagem para o sistema de arquivos;
6. Instalar o Kernel.

Uma vez que o processo tenha sido concluído será criado um diretório chamado *output* com o seguinte conteúdo:
build host images staging stamps target toolchain

Para a aula, não precisamos esperar a conclusão do processo. Podem ser utilizados arquivos gerados previamente com configurações similares as vistas nesse tutorial. Para isto, basta baixar o arquivo "rootfs.tar.gz" e copiar os arquivos para o cartão. Esse arquivo geralmente fica no diretório "output/images" após a execução do BuildRoot.

Copiar o sistema de arquivos para o cartão:
tar -C /media/rootfs -xvzf rootfs.tar.gz

Serão necessárias algumas modificações. Por padrão, o usuário root vem sem password. Definir um password para esse usuário. Esse password deverá ser chamado rpi(NUMERO_DO_GRUPO), conforme o exemplo abaixo para o grupo 00:
*CRYPTEDPASS=$(perl -e 'print crypt("**rpi00**","salt")')*

sed -i -e "s#^root:[^:]:#root:$CRYPTEDPASS:#" /media/rootfs/etc/shadow*

Ao inicializar a Raspberry Pi, queremos montar a primeira partição do cartão como */boot*. Para tanto, definir um ponto de montagem para essa partição:
install -d -m 0755 /media/rootfs/boot
echo "/dev/mmcblk0p1 /boot vfat defaults 0 0" >> /media/rootfs/etc/fstab

Baixar o arquivo "zImage", que contem o Kernel compilado. Esse arquivo geralmente fica no diretório "output/images" após a execução do BuildRoot. Copiar o Kernel gerado para a partição de boot:
cp zImage /media/boot/kernel.img

Copiar o firmware para a partição de boot. Para tanto, baixar o arquivo *firmware.zip* e copiar seu conteúdo:
unzip firmware.zip
cp firmware/bootcode.bin /media/boot
cp firmware/start.elf /media/boot
cp firmware/fixup.dat /media/boot

Finalmente, adicionar uma linha com comandos de inicialização para a Raspberry Pi. Lembrar de colocar o número do grupo nas configurações de IP, conforme destacado abaixo:
*echo "dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 elevator=deadline rootwait ip=::::rpi**00**::dhcp root=/dev/mmcblk0p2 rootfstype=ext4" > /media/boot/cmdline.txt*

Salvar as modificações e desmontar o cartão.
sync
umount /media/boot
umount /media/rootfs

OBS: É extremamente importante executar o comando *sync* antes de desmontar o cartão. Ele garantirá que todos os arquivos copiados para o cartão foram de fato armazenados em sua memória flash e nada ficou no buffer.

Executando o Linux embarcado a partir na Raspberry Pi:

Colocar o cartão na Raspberry Pi, contectá-la a rede usando um cabo ethernet e conectá-la a alimentação (cabo USB). A placa deverá ligar e em alguns segundos estará executando o Linux com um servidor SSH habilitado. Para obter o IP que foi designado a placa, executar o seguinte comando e buscar pelo IP atribuído de acordo com o MAC da placa:
nmap -sP 10.32.143.0/24

Uma informação semelhante a apresentada abaixo será recebida:
```
Host 10.32.143.111 is up (0.00035s latency).
MAC Address: BC:AE:C5:C3:16:93 (Unknown)
```

Conectar como root ao ip designado a placa (usar a senha gerada anteriormente):
ssh root@10.32.143.111

Se tudo deu certo, você deve ter acesso ao Linux embarcado na Raspberry Pi!

Esse Linux será a base para o desenvolvimento de atividades no decorrer do semestre. Nunca esqueça de antes de desconectar a placa da fonte de alimentação, desligar o sistema através do comando:
halt

Caso isso não seja feito, o sistema de arquivos poderá ser corrompido e todo processo de geração do Linux deverá ser reelaborado. A partir desse ponto, seu grupo deverá seguir utilizando sempre a mesma placa e cartão.

"Crosscompilando":

Para gerar compilar programas que possam ser executados na Raspberry Pi, podemos utilizar o toolchain gerado pelo próprio BuildRoot. Para tanto, pode ser usado o ambiente gerado no seguinte diretório:
~/buildroot/buildroot-2013.05/output/host/usr/bin

Como não esperamos a geração desse diretório, deve-se usar o que está disponível site da disciplina, com o mesmo conteúdo. Baixar o arquivo "crosscompile.tar.gz" e extraí-lo para o home.
tar -C ~ -xvzf crosscompile.tar.gz

Mapear o diretório para usar o toolchain:
export PATH=$PATH:~/crosscompile/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/crosscompile/lib

Criar um programa de teste:
nano hello.c

Inserir o seguinte código:

```
#include <stdio.h>

int main(void)

{

  printf("Hello, cross-compilation world !\n");

  return 0;

}
```

Salvar o programa, sair e compilar:
arm-buildroot-linux-uclibcgnueabi-gcc teste.c -o teste

Enviar o programa gerado para a placa:
scp teste root@(IP_ATRIBUIDO_A_PLACA):./

Conectar-se a placa e executar o programa. Se tudo deu certo, a seguinte mensagem será impressa:
"Hello, cross-compilation world !"