

Compilando e Gravando o Linux

Neste tutorial irá ser visto como compilar e gravar corretamente o kernel do linux que será bastante usado na disciplina e na vida profissional. O kernel do linux é exatamente o que a tradução sugere, é o nucleo do sistema operacional, ou seja, a parte mais importante de um sistema embarcado. Para a disciplina, é bastante necessário que tenha esse conceito em mente.

Baixando o Kernel

O pacote com o kernel do linux pode ser encontrado no github da propria beaglebone, então usaremos o comando "git" para baixar a versão mais nova do U-Boot. E então entraremos na pasta para começar a configuração.

```
$ git clone git://github.com/beagleboard/linux.git  
$ cd linux
```

Configurando o Kernel

Após baixado e acessar a pasta, é preciso informar ao git qual a versão deseja usar, neste caso, será feito com a versão mais nova, ou seja, a versão 4.4, então insira o seguinte comando para usar o branch da versão:

```
$ git checkout 4.4
```

Com isso, agora podemos pré-configurar o kernel do linux para a beaglebone black, que é a placa que estamos usando.

```
$ make ARCH=arm CROSS_COMPILE=arm-linux- bb.org_defconfig
```

Compilando o Kernel

Agora o kernel está pronto para ser compilado. Então execute o seguinte comando:

```
$ make ARCH=arm CROSS_COMPILE=arm-linux- -j4
```

Este processo pode demorar algum tempo, dependendo de computador para computador. Após o kernel ser compilado, gera uma versão de uma imagem com o device tree. O device tree serve como os drivers no sistema operacional Windows. A variável de ambiente LOADADDR serve para indicar em qual posição de memória está o device tree.

```
$ make ARCH=arm CROSS_COMPILE=arm-linux- uImage dtbs LOADADDR=0  
x80008000 -j4
```

Após gerar o arquivo "uImage", veja usando o comando "ls", se o arquivo realmente está no diretório "arch/arm/boot/", o resultado esperado é o seguinte:

```
-rw-r--r-- 1 robertcabral robertcabral 7846304 Sep 18 12:31 uImage
```

Verifique também se o arquivo "am335x-boneblack.dtb" está no diretório "arch/arm/boot/dts". Com esses dois arquivos gerados, copie eles para seu servidor TFTP.

```
-rw-r--r-- 1 robertcabral robertcabral 54491 Sep 18 12:10 arch/arm/
      boot/dts/am335x-boneblack.dtb
$ cp arch/arm/boot/dts/am335x-boneblack.dtb /tftpboot/
$ cp arch/arm/boot/uImage /tftpboot/
```

Rodando o kernel gerado

Após a geração das imagens e a copia para o servidor TFTP, podemos então rodar essas imagens usando a placa. Use o bootloader no cartão SD, como foi feito na ultima prática, pois vai lhe poupar bastante tempo. Inicie o bootloader pelo cartão SD e insira os seguintes comandos:

```
U-BootEC$ setenv serverip 192.168.0.1
U-BootEC$ setenv ipaddr 192.168.0.2
```

Repare que a configuração feita foi basicamente setar os ip's, tanto do servidor, como da propria placa. Não esqueça porém que suas configurações de rede podem ser diferentes dessa, como por exemplo o ip do servidor ser "10.4.2.1", nesse caso a variavel "serverip" iria receber o ip "10.4.2.1". Além disso, o ip da placa deve está na mesma faixa de ip's do servidor, nesse caso a variavel "serverip" deve ser igual a "ipaddr", mudando somente o ultimo digito.

Feito isso, o próximo passo é configurar a variável "bootargs", que é a responsável por passar informações para o Linux. Seu conteúdo é automaticamente passado para o kernel Linux como argumentos de inicialização. Como vamos usar a porta serial como console, então a configuração da variável é a seguinte:

```
U-BootEC$ setenv bootargs "console=ttyS0,115200,n8"
```

E por fim, é preciso baixar as imagens que estão no servidor TFTP para regiões de memória da placa. Então, para não precisarmos fazer as mesmas configurações mais de uma vez, podemos configurar a variável "bootcmd", que é basicamente a variável que guarda o comando que a placa vai executar caso nenhuma tecla seja pressionada para parar a inicialização, então basicamente essa variável deve ser configurada da seguinte forma:

```
U-BootEC$ setenv bootcmd "tftp 0x80F80000 am335x-boneblack.dtb; tftp
      0x80007FC0 uImage; bootm 0x80007FC0 - 0x80F80000;"
```

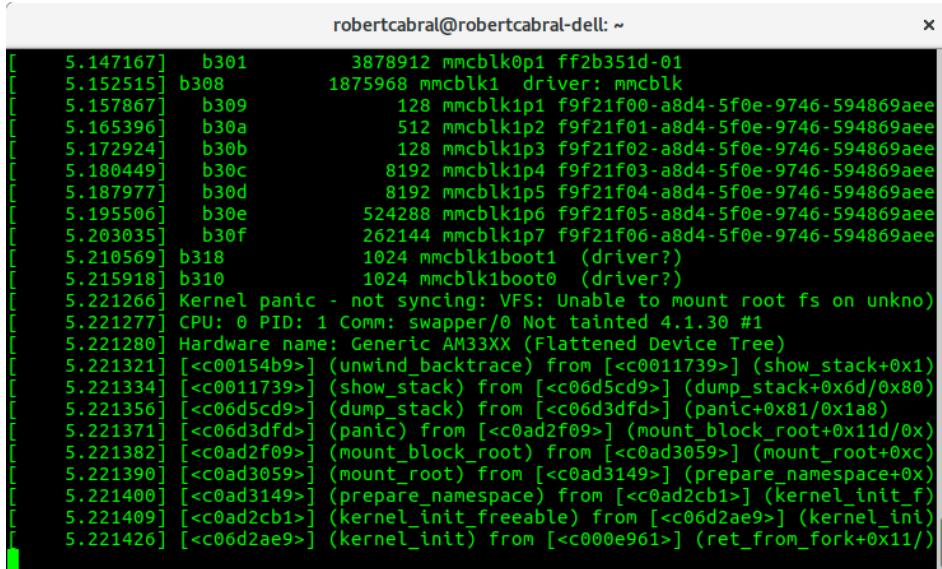
Com todas as variáveis configuradas, podemos guardar as alterações feitas nas variaveis de ambiente(serverip, ipaddr, bootargs, bootcmd), poupano tempo. Não esqueça que todos esses passos devem ser feitos usando o bootloader gerado anteriormente e usando o cartão SD.

```
U-BootEC$ saveenv
```

Depois de seguir todos os passos corretamente, já podemos rodar o kernel compilado usando o seguinte comando:

```
U-BootEC$ boot
```

E assim o resultado esperado deve ser o seguinte:



```

robertcabral@robertcabral-dell: ~
[ 5.147167] b301      3878912 mmcblk0p1 ff2b351d-01
[ 5.152515] b308      1875968 mmcblk1 driver: mmcblk
[ 5.157867] b309      128 mmcblk1p1 f9f21f00-a8d4-5f0e-9746-594869aee
[ 5.165396] b30a      512 mmcblk1p2 f9f21f01-a8d4-5f0e-9746-594869aee
[ 5.172924] b30b      128 mmcblk1p3 f9f21f02-a8d4-5f0e-9746-594869aee
[ 5.180449] b30c      8192 mmcblk1p4 f9f21f03-a8d4-5f0e-9746-594869aee
[ 5.187977] b30d      8192 mmcblk1p5 f9f21f04-a8d4-5f0e-9746-594869aee
[ 5.195506] b30e      524288 mmcblk1p6 f9f21f05-a8d4-5f0e-9746-594869aee
[ 5.203035] b30f      262144 mmcblk1p7 f9f21f06-a8d4-5f0e-9746-594869aee
[ 5.210569] b318      1024 mmcblk1boot1 (driver?)
[ 5.215918] b310      1024 mmcblk1boot0 (driver?)
[ 5.221266] Kernel panic - not syncing: VFS: Unable to mount root fs on unkno)
[ 5.221277] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 4.1.30 #1
[ 5.221280] Hardware name: Generic AM33XX (Flattened Device Tree)
[ 5.221321] [<c00154b9>] (unwind_backtrace) from [<c0011739>] (show_stack+0x1)
[ 5.221334] [<c0011739>] (show_stack) from [<c06d5cd9>] (dump_stack+0x6d/0x80)
[ 5.221356] [<c06d5cd9>] (dump_stack) from [<c06d3df0>] (panic+0x81/0x1a8)
[ 5.221371] [<c06d3df0>] (panic) from [<c0ad2f09>] (mount_block_root+0x11d/0x)
[ 5.221382] [<c0ad2f09>] (mount_block_root) from [<c0ad3059>] (mount_root+0xc)
[ 5.221390] [<c0ad3059>] (mount_root) from [<c0ad3149>] (prepare_namespace+0x)
[ 5.221406] [<c0ad3149>] (prepare_namespace) from [<c0ad2cb1>] (kernel_init_f)
[ 5.221409] [<c0ad2cb1>] (kernel_init_freeable) from [<c06d2ae9>] (kernel_init)
[ 5.221426] [<c06d2ae9>] (kernel_init) from [<c000e961>] (ret_from_fork+0x11/)
```

Figura 1: Tela obtida.

Perceba que o kernel retorna uma mensagem que entrou em "pânico", isso ocorreu porque ele não encontrou o sistema de arquivo para executar. Veja a próxima seção para corrigir esse erro.

Sistema de arquivo

Um sistema de arquivos é um conjunto de estruturas lógicas que permite o sistema operacional controlar o acesso a um dispositivo de armazenamento. No caso o seu próprio computador tem um sistema de arquivos, que fica acessando dados do disco rígido. Caso o sistema operacional não encontre um sistema de arquivos, ele não pode executar e entrará em pânico.

Criando um sistema de arquivos

Para criar o nosso próprio rootfs(sistema de arquivos), usaremos o Busybox, que é um projeto open source que nos possibilita o uso sem custos. O primeiro passo é baixar o busybox, que pode ser encontrado tanto no site <https://busybox.net/>, como no seguinte comando:

```
$ wget https://busybox.net/downloads/busybox-1.27.2.tar.bz2
```

A ultima versão estavel é a versão 1.27.2, então essa que usaremos para prosseguir. Agora descompacte o arquivo baixado e entre na pasta:

```
$ tar xfv busybox-1.27.2.tar.bz2
$ cd busybox-1.27.2/
```

E então entre nas configurações do busybox e accese o menu "Busybox Settings" e na categoria "Build Options", mude a opção "Cross Compiler prefix" e adicione o compilador gerado anteriormente. Devendo ter o seguinte resultado:

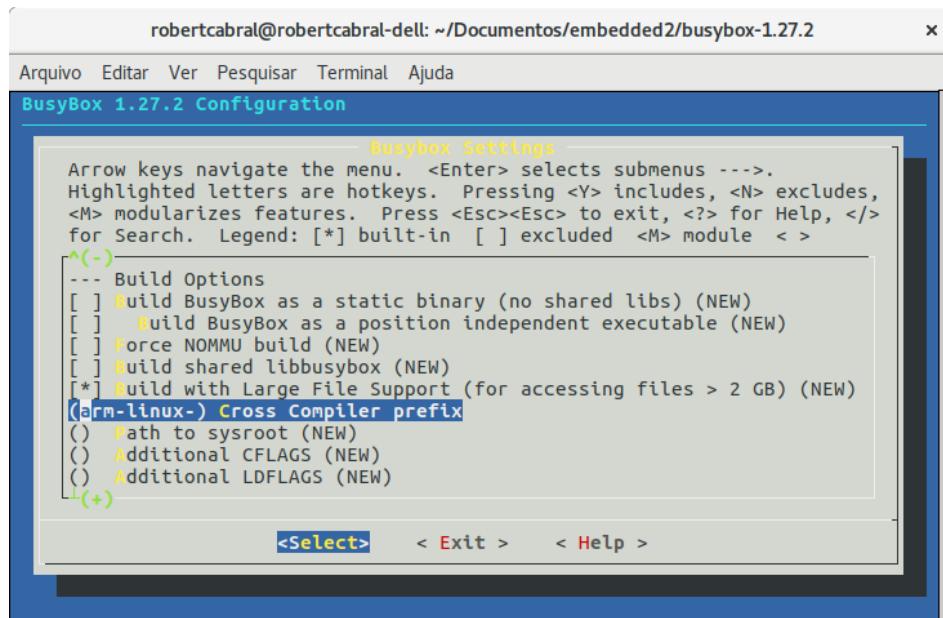


Figura 2: Configuração do compilador Busybox.

Salve e saia, e então compile o busybox usando o seguinte comando:

```
$ make busybox -j4
```

Depois que a compilação terminar, verifique se o binário foi gerado corretamente, usando o comando file.

```
$ file busybox
busybox: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV),
dynamically linked, interpreter /lib/ld-linux.so.3, for GNU/Linux
3.2.72, stripped
```

Com o binário gerado, agora podemos criar o rootfs, que é onde vai ficar todas as informações que serão usadas no linux. Agora execute o comando a seguir para que o rootfs seja gerado com todas as aplicações e ferramentas habilitadas no menu de configuração. Não esqueça que deve tá na pasta do busybox ainda.

```
$ make CONFIG_PREFIX=/home/robertcabral/Documentos/embedded2/rootfs
install
```

Onde deve usar o um caminho próprio no seu computador. Não se preocupe em criar a pasta, pois o busybox se encarregará disso.

Como o Busybox foi linkado dinamicamente, precisamos copiar as bibliotecas do sistema para o rootfs de forma que possamos executar o Busybox na placa. Para isso entre na pasta do compilador que gerou anteriormente usando o crosstool-ng e siga os seguintes passos(lembre-se que deve substituir os caminhos para os caminhos corretos para seu computador).

```
$ cd ~/Documentos/monitoria/compiler/
$ cd arm-cortex_a8-linux-gnueabi/
```

E agora copie a pasta lib/ para a pasta onde gerou o rootfs:

```
$ cp -av lib/ ~/Documentos/embedded2/rootfs/  
$ chmod u+w ~/Documentos/embedded2/rootfs/lib
```

Agora o seu rootfs já está quase pronto. Entre na pasta do rootfs e crie as pastas dev/ proc/ sys/ e etc/.

```
$ cd ~/Documentos/embedded2/rootfs/  
$ mkdir dev/ proc/ sys/ etc/
```

E então neste momento o rootfs só precisa de mais uma unica coisa, que são os scripts de boot. O primeiro processo que o kernel executará será o /sbin/init Este processo é disponibilizado pelo Busybox e segue o padrão System V Init. O /sbin/init procura e executa os comandos descritos no arquivo de configuração /etc/inittab. Então crie este arquivo:

```
$ gedit etc/inittab
```

E adicione o seguinte script:

```
# run /bin/sh on ttys0  
ttys0::respawn:/sbin/getty -L 115200 ttys0 vt100
```

Perceba que no inittab estamos iniciando a aplicação getty, que irá executa a aplicação de terminal (/bin/sh) na conexão serial ttys0. Ou seja, quando você iniciar o target, terá acesso ao terminal para a execução de comandos.

Instalando e configurando o servidor NFS

Um servidor NFS(Network File System) é um protocolo que permite acesso remoto a um sistema de arquivos através da rede.

Para instalar o servidor NFS, execute o seguinte comando:

```
$ sudo apt-get install nfs-kernel-server
```

Uma vez instalado, precisamos configura-lo para termos acesso através da placa. Então para isso, abra o arquivo /etc/exports.

```
$ sudo gedit /etc/exports
```

E adicione a seguinte linha no final do arquivo:

```
/home/robertcabral/Documentos/embedded2/rootfs 192.168.0.2(rw,  
no_root_squash,no_subtree_check)
```

Onde o primeiro argumento é a pasta onde fica o rootfs, o segundo simplesmente vai liberar acesso dessa pasta para o ip da placa(192.168.0.2) e no ultimo argumento o parâmetro rw possibilita o acesso de escrita e leitura, o parâmetro no_root_squash possibilita o acesso com permissões de root e o parâmetro no_subtree_check desabilita a verificação da localidade do arquivo em cada acesso para aumentar a velocidade e a confiabilidade no acesso.

Abra o arquivo /etc/hosts.allow.

```
$ sudo gedit /etc/hosts.allow
```

E adicione a seguinte linha no final do arquivo:

```
rpcbind mountd nfsd statd lockd rquotad : 192.168.0.2
```

Abra o arquivo /etc/hosts.deny.

```
$ sudo gedit /etc/hosts.deny
```

E adicione a seguinte linha no final do arquivo:

```
rpcbind mountd nfsd statd lockd rquotad : ALL
```

E então reinicie o servidor com o seguinte comando:

```
$ sudo service nfs-kernel-server restart
```

Configurando o servidor NFS na placa

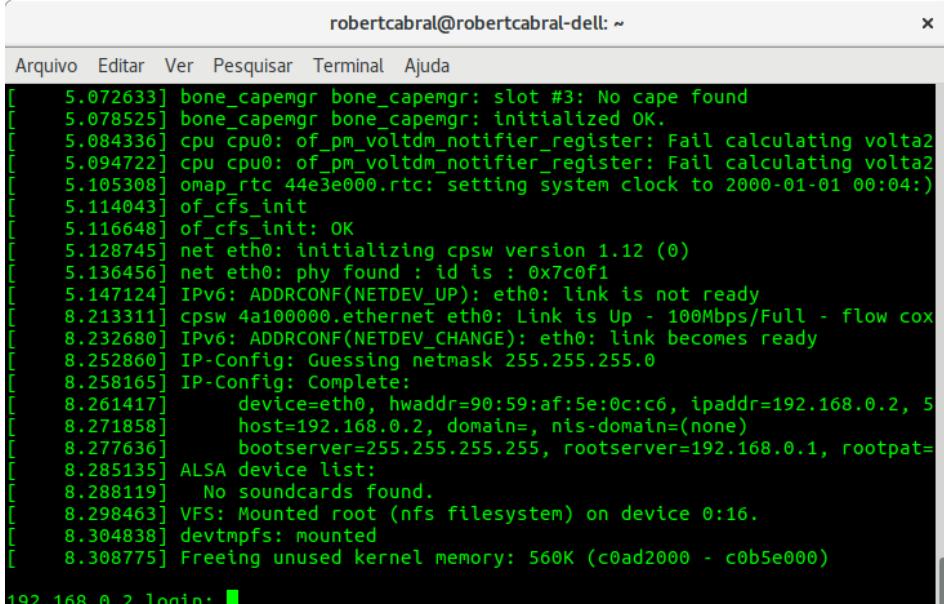
Para configurar o target para acessar o rootfs via NFS, precisamos alterar a variável de ambiente bootargs do U-Boot, passando os parâmetros de sistema de arquivo por NFS. Usando o mesmo bootloader gerado execute os seguintes comandos:

```
U-BootEC$ setenv argsnfs root=/dev/nfs rw nfsroot=${serverip}:/home/robertcabral/Documentos/embedded2/rootfs rootwait
U-BootEC$ setenv consolecfg console=ttyS0,115200,n8
U-BootEC$ setenv ipcfg ip=192.168.0.2
U-BootEC$ setenv bootargs ${consolecfg} ${ipcfg} ${argsnfs}
U-BootEC$ saveenv
```

Modifique o caminho da pasta "rootfs"para o caminho que ela está no seu computador. Com isso já pode dar o comando "boot", para iniciar o kernel.

```
U-BootEC$ boot
```

O resultado esperado é o seguinte:



```

robertcabral@robertcabral-dell: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
[ 5.072633] bone_capemgr bone_capemgr: slot #3: No cape found
[ 5.078525] bone_capemgr bone_capemgr: initialized OK.
[ 5.084336] cpu cpu0: of_pm_voltdm_notifier_register: Fail calculating volta2
[ 5.094722] cpu cpu0: of_pm_voltdm_notifier_register: Fail calculating volta2
[ 5.105308] omap_rtc 44e3e000 rtc: setting system clock to 2000-01-01 00:04:)
[ 5.114043] of_cfs_init
[ 5.116648] of_cfs_init: OK
[ 5.128745] net eth0: initializing cpsw version 1.12 (0)
[ 5.136456] net eth0: phy found : id is : 0x7c0f1
[ 5.147124] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[ 8.213311] cpsw 4a100000.ethernet eth0: Link is Up - 100Mbps/Full - flow cox
[ 8.232680] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 8.252860] IP-Config: Guessing netmask 255.255.255.0
[ 8.258165] IP-Config: Complete:
[ 8.261417]           device=eth0, hwaddr=90:59:af:5e:0c:c6, ipaddr=192.168.0.2, 5
[ 8.271858]           host=192.168.0.2, domain=, nis-domain=(none)
[ 8.277636]           bootserver=255.255.255.255, rootserver=192.168.0.1, rootpath=
[ 8.285135] ALSA device list:
[ 8.288119]           No soundcards found.
[ 8.298463] VFS: Mounted root (nfs filesystem) on device 0:16.
[ 8.304838] devtmpfs: mounted
[ 8.308775] Freeing unused kernel memory: 560K (c0ad2000 - c0b5e000)

192.168.0.2 login: 

```

Figura 3: Servidor NFS rodando na placa.

Perceba que nenhuma senha consegue se autenticar no linux que você gerou, para corrigir esse pequeno problema, é preciso criar dois arquivos na pasta do rootfs, onde são eles o arquivo etc/passwd e o arquivo etc/shadow, que um fica a lista de usuários e o outro as senhas dos usuários. Nesse caso crie o arquivo etc/passwd.

```
$ gedit ~/Documentos/embedded2/rootfs/etc/passwd
```

E adicione a seguinte linha:

```
root:x:0:0:root:/root:/bin/sh
```

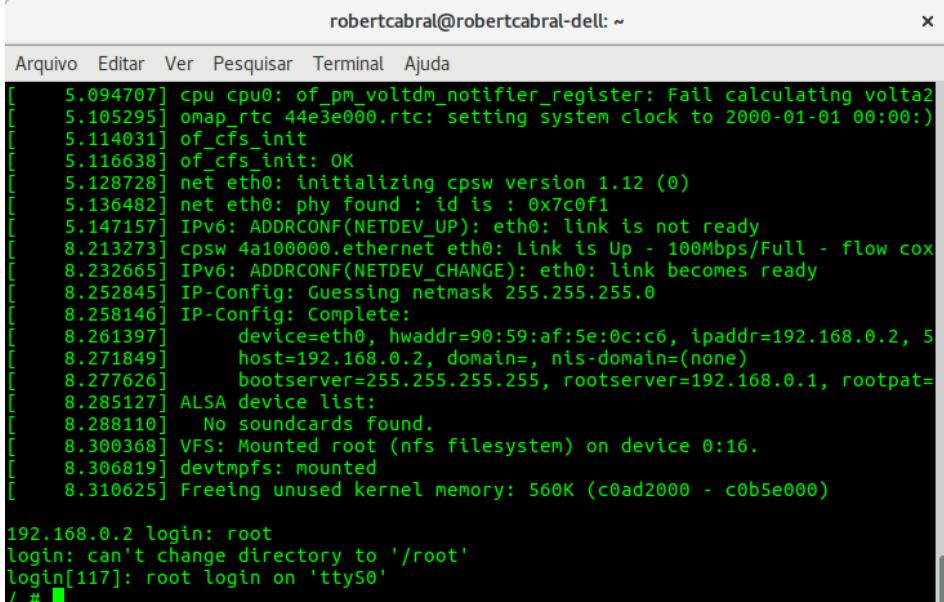
E então, crie o arquivo etc/shadow.

```
$ gedit ~/Documentos/embedded2/rootfs/etc/shadow
```

E adicione a seguinte linha:

```
root::17409:0:99999:7:::
```

Dessa forma, "root"será um usuário sem senha. Assim já é possível acessar o sistema com esse usuário.



```

robertcabral@robertcabral-dell: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
[ 5.094707] cpu cpu0: of_pm_voltm_notifier_register: Fail calculating volta2
[ 5.105295] omap_rtc 44e3e000.rtc: setting system clock to 2000-01-01 00:00:00)
[ 5.114031] of_cfs_init
[ 5.116638] of_cfs_init: OK
[ 5.128728] net eth0: initializing csw version 1.12 (0)
[ 5.136482] net eth0: phy found : id is : 0x7c0f1
[ 5.147157] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[ 8.213273] csw 4a10000.ethernet eth0: Link is Up - 100Mbps/Full - flow cox
[ 8.232665] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 8.252845] IP-Config: Guessing netmask 255.255.255.0
[ 8.258146] IP-Config: Complete:
[ 8.261397]           device=eth0, hwaddr=90:59:af:5e:0c:c6, ipaddr=192.168.0.2, 5
[ 8.271849]           host=192.168.0.2, domain=, nis-domain=(none)
[ 8.277626]           bootserver=255.255.255.255, rootserver=192.168.0.1, rootpath=
[ 8.285127] ALSA device list:
[ 8.288110]   No soundcards found.
[ 8.300368] VFS: Mounted root (nfs filesystem) on device 0:16.
[ 8.306819] devtmpfs: mounted
[ 8.310625] Freeing unused kernel memory: 560K (c0ad2000 - c0b5e000)

192.168.0.2 login: root
login: can't change directory to '/root'
login[117]: root login on 'ttys0'
/ #

```

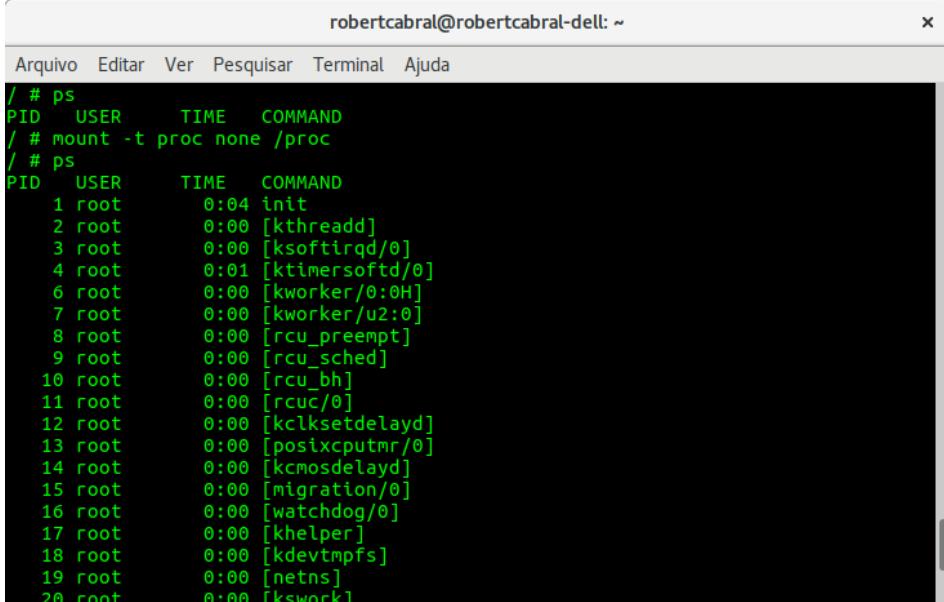
Figura 4: Usuário logado na placa.

Melhorando o rootfs

Ao tentar executar o comando ps para listar os processos em execução, o linux não encontra processos, isso porque o comando ps utiliza o sistema de arquivo virtual procfs, que deve estar montado no diretório /proc. Então precisamos montar o procfs, com o seguinte comando:

```
# mount -t proc none /proc
```

E então, executando o comando ps, pode ser então visualizado os processos. Não esqueça que esses passos devem ser feitos usando o linux na placa.



```

robertcabral@robertcabral-dell: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
/ # ps
PID  USER      TIME  COMMAND
/ # mount -t proc none /proc
/ # ps
PID  USER      TIME  COMMAND
  1 root      0:04 init
  2 root      0:00 [kthreadd]
  3 root      0:00 [ksoftirqd/0]
  4 root      0:01 [ktimersoftd/0]
  6 root      0:00 [kworker/0:0H]
  7 root      0:00 [kworker/u2:0]
  8 root      0:00 [rcu_prempt]
  9 root      0:00 [rcu_sched]
 10 root      0:00 [rcu_bh]
 11 root      0:00 [rcuc/0]
 12 root      0:00 [kclksetdelayd]
 13 root      0:00 [posixcpumr/0]
 14 root      0:00 [kcmosdelayd]
 15 root      0:00 [migration/0]
 16 root      0:00 [watchdog/0]
 17 root      0:00 [khelper]
 18 root      0:00 [kdevtmpfs]
 19 root      0:00 [netns]
 20 root      0:00 [kswork]

```

Figura 5: Processos rodando no linux gerado.

Outro sistema de arquivo virtual importante no Linux é o sysfs, montado no diretório /sys. O /sys é muito importante para controlar dispositivos de hardware usados na placa, como por exemplo acender led's. Então monte também essa partição.

```
# mount -t sysfs none /sys
```

E então, agora podemos acender um led. Vamos acender um dos leds de usuário, que vem na própria placa. Para isso, escreva "1" no arquivo "brightness".

```
# echo 1 > /sys/class/leds/beaglebone\:green\:usr1/brightness
```

Dessa forma, o led 1 do usuário, perceba na figura 6 o led apagado e após o comando, o led acesso.

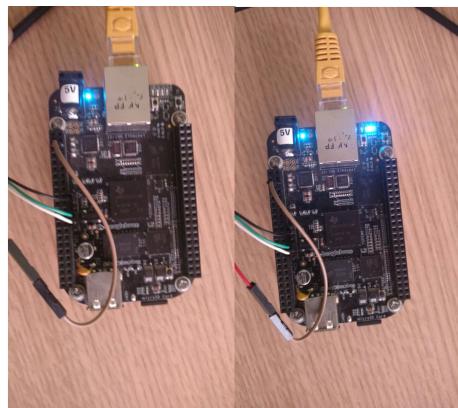


Figura 6: Led acesso.

Configurando um servidor web no linux

O uso da web em sistemas embarcados estão cada vez mais crescentes. Então vamos criar nosso próprio servidor web no linux gerado. Para isso crie uma pasta chamada "www" no diretório do seu rootfs:

```
$ cd ~/Documentos/embedded2/rootfs
$ mkdir www/
```

E então crie um arquivo chamado "index.html" e insira um código simples qualquer.

```
1 <HTML>
2   <HEAD> <TITLE>Embarcados II</TITLE> </HEAD>
3   <BODY>
4     <H1>Web Server BBB!</H1>
5   </BODY>
6 </HTML>
```

Então, acesse o linux da placa e inicie o serviço com o seguinte comando:

```
# httpd -h /www/
```

Acesse o navegador do seu computador no endereço IP da sua placa, no meu caso acessei o endereço "192.168.0.2":



Figura 7: Servidor WEB.

Módulos no Kernel

Em computação, um módulo carregável do núcleo é um arquivo objeto que contém código para estender o núcleo em execução, ou o chamado núcleo base, de um sistema operacional.

Compilando e instalando os módulos

Abra um terminal e entre no diretório do código-fonte do kernel e então entre no menu de configurações:

```
$ cd ~/Documentos/monitoria/linux/linux/  
$ make ARCH=arm menuconfig
```

E então acesse o menu "Device Drivers" e o submenu "SCSI device support" e marque com um "M", que é de módulo.

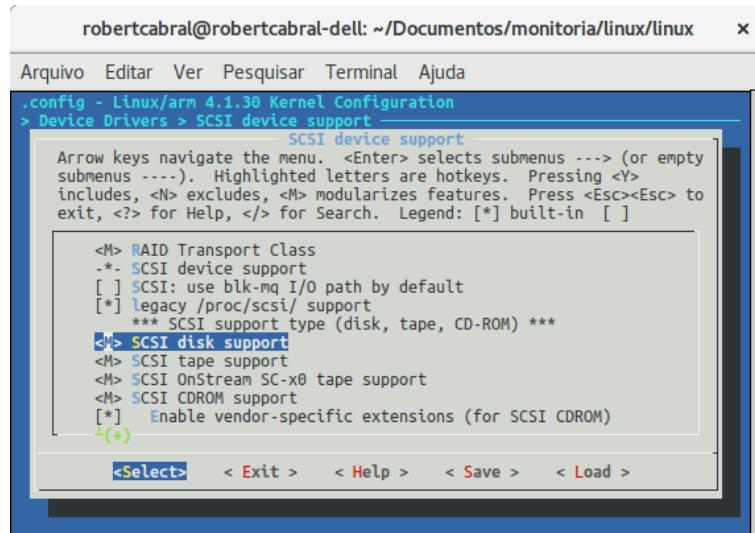


Figura 8: Modulo configurado.

E agora acesse o menu "Device Drivers" e o submenu "USB support" e marque com um "M", o suporte a dispositivos de armazenamento USB.

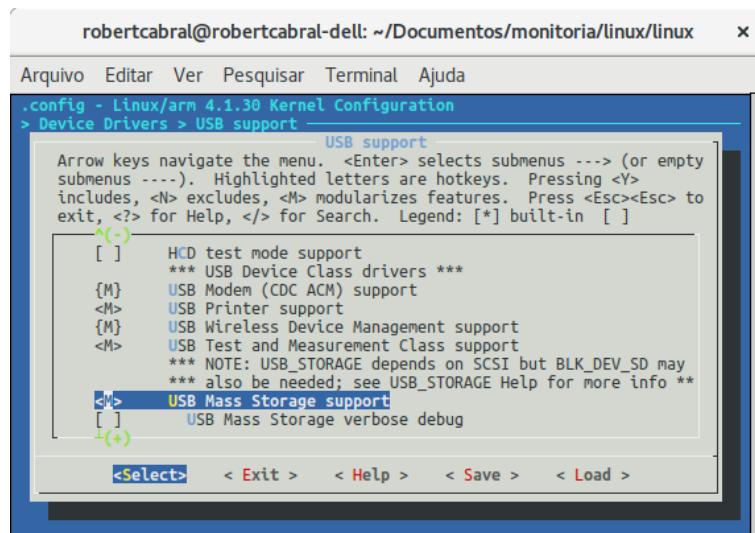


Figura 9: Modulo configurado.

Salve e saia, e então compile:

```
$ make ARCH=arm CROSS_COMPILE=arm-linux- modules -j4
```

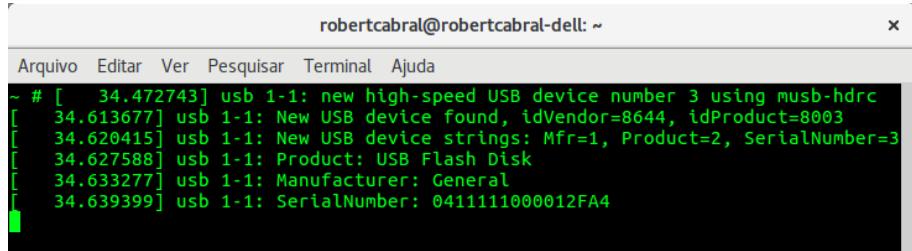
E agora instale os módulos no rootfs.

```
$ make ARCH=arm INSTALL_MOD_PATH=~/Documentos/embedded2/rootfs modules_install
```

Perceba que o kernel instalou os módulos em lib/modules/

```
$ ls ~/Documentos/embedded2/rootfs/lib/modules/*
```

Insira um pendrive no linux gerado, perceba que o linux não conseguiu identificar o pendrive:



```
robertcabral@robertcabral-dell: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
~ # [ 34.472743] usb 1-1: new high-speed USB device number 3 using musb-hdrc
[ 34.613677] usb 1-1: New USB device found, idVendor=8644, idProduct=8003
[ 34.620415] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 34.627588] usb 1-1: Product: USB Flash Disk
[ 34.633277] usb 1-1: Manufacturer: General
[ 34.639399] usb 1-1: SerialNumber: 041111000012FA4
```

Figura 10: USB não reconhecido.

Inserindo módulos

Para inserir o módulo com os seguintes comandos:

```
# insmod /lib/modules/$(uname -r)/kernel/drivers/usb/storage/usb-
storage.ko
# modprobe usb-storage
```

Com o comando modprobe, todas as dependências do módulo usb-storage também foram inseridas. Você pode listar os módulos carregados com o seguinte comando:

```
# lsmod
```

Não esqueça que os comandos de montar as partições virtuais sysfs e proc, que estão descritos na seção "Melhorando o rootfs".

Com o pendrive ainda inserido, veja qual o nome do dispositivo para o linux:

```
# ls /dev/sd*
```

Montando um pendrive

Depois de inserir os módulos de USB, podemos usar um dispo USB para fazer testes. Para isso, é preciso criar a pasta /media.

```
# mkdir /media
```

E então, crie o arquivo /etc/fstab, e insira no final da linha a seguinte linha:

```
UUID=84EB-383B /media fat32 defaults 0 1
```

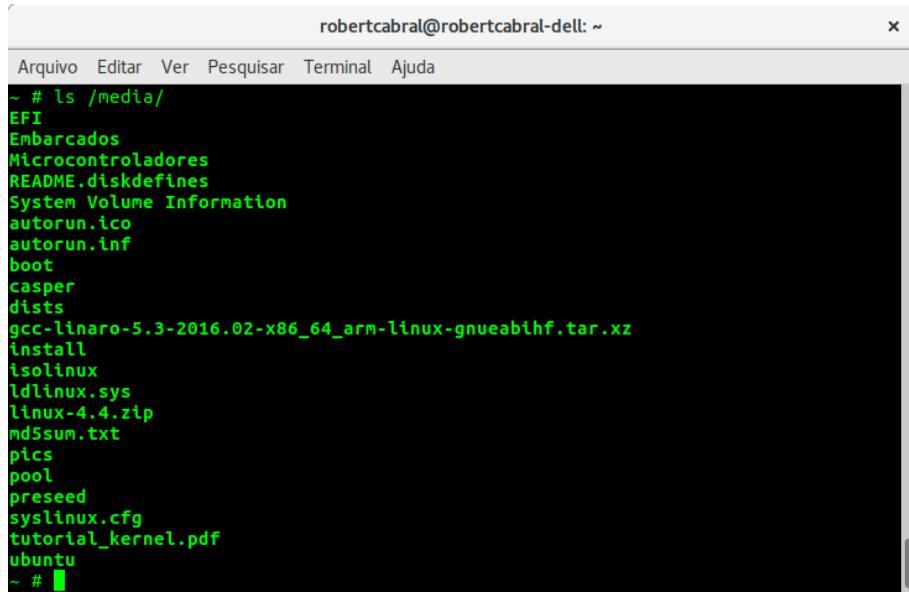
Onde o parametro UUID pode ser consultado com o seguinte comando:

```
# blkid /dev/sda1
/dev/sda1: LABEL="UBUNTU-GNOM" UUID="84EB-383B"
```

Use o nome da partição que o seu pendrive está. Perceba que temos o nome do pendrive e o UUID. Copie o UUID e substitua no arquivo /etc/fstab. Agora podemos simplesmente montar o dispositivo, use esse comando:

```
# mount /dev/sda1 /media
# ls /media
```

Com isso, temos o nosso pendrive montado e funcionando.



A screenshot of a terminal window titled "robertcabral@robertcabral-dell: ~". The window has a menu bar with "Arquivo", "Editar", "Ver", "Pesquisar", "Terminal", and "Ajuda". Below the menu is a command-line interface. The user has run the command "ls /media" which lists several directories and files. The output is as follows:

```
robertcabral@robertcabral-dell: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
~ # ls /media/
EFI
Embarcados
Microcontroladores
README.diskdefines
System Volume Information
autorun.ico
autorun.inf
boot
casper
dists
gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf.tar.xz
install
isolinux
ldlinux.sys
linux-4.4.zip
md5sum.txt
pics
pool
preseed
syslinux.cfg
tutorial_kernel.pdf
ubuntu
~ #
```

Figura 11: USB montado.

Caso queira desmonta-lo use o seguinte comando:

```
# umount /dev/sda1
```

Removendo os módulos

Para remover os módulos use o seguinte comando:

```
# modprobe -r usb-storage
```