Tutorial BuildRoot

Programação de Periféricos – CC – FACIN – PUCRS

Configuração do Ambiente de Trabalho:

- Criar um diretório de trabalho: Ex: mkdir ~/buildroot
- 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:

row keys navigate lectes a feature, gend: [*] feature	the menu, <enter> selects submenus>. Highlighted letters are hotkeys. Pressing <y> while <n> will exclude a feature. Press <esc><esc> to exit, <>> for Help, for Search. ; is selected [] feature is excluded</esc></esc></n></y></enter>
	Target Architecture (1336)> Target Architecture Variant (1586)> Tould options> Touldoptions> Toulchain> Tystem configuration> Tystem configuration> Tackage Selection for the target> Most utilities> Fackage Selection for the target> Most utilities> Toulcoders> Ternel> Ternel> Ternel> Ternel>

Meu Primeiro Linux embarcado:

Configurar a arquiteutura alvo de acordo com a Raspberry Pi:

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

row keys navigate the on lectes a feature, while gend: [*] feature is selv	u. ≪Enter> su N> will exclus icted [] feat	elects sub de a featu ture is ex	nenus>. re. Press < cluded	iguration Highlighted Esc> <esc> to</esc>	letters are exit, f	hotkeys. Pressing <y> or Help, for Search.</y>
	riget Architect riget Architect ild aptions olchain	ture (ARM ture Vario on for the > ts> > otions	(little endi nt (nrelif/0) target	an))> 2(+s)> >		
	-					

Configurar opções do BuildRoot:

Selecionar a opção Build Op	<i>tions</i> 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:

rrow keys (electes a egend: [*]	maild options avigate the menu. <enter> selects submenus>. Highlighted letters are hotkeys. Pressing <y> eature, while <n> will exclude a feature. Press <esc> to exit, <? > for Help, for Search. feature is selected [] feature is excluded</esc></n></y></enter>
	<pre>commands> (\$(CONFIG DIR)/defconfig) Location to save buildrnot config (NEW) (\$(HOME)/buildroot/dl) bowhload dir (\$(BASE DIR)/ost) Host dir (NEW) Mirrors and Download locations> (8) Number of jobs to run simultaneously (8 for auto) (NEW) [*] (mable compiler cache [*(HOWE)/buildroot/ccache) Compiler cache location [\$ Show packages that are deprecated or obsolete (NEW) [build packages with debugging symbols (NEW) strip command for binaries on target (strip)> () executables that should not be stripped (NEW) [directories that should be skipped when stripping (NEW) gcc optimization level (optimize for size)> </pre>
	gcc optimization level (optimize for size)>

Configurar opções do Toolchain:

Selecionar a opção Toolchain do men	u.
Kernel Headers:	
GCC compiler Version:	
Additional gcc options:	
Configurações do GCC:	
Purge unwanted locales	YES

Linux 3.9.x kernel headers GCC 4.8.x --with-float=hard --with-fpu=vfp

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) suppor	t:	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 hostnamerpi00 (NUMERO DO GRUPO)System bannerWelcome to Raspberry Pi!/dev managementDynamic using mdevPort to run a getty (login prompt) ontty1Baudrate to use38400

electes a egend: [*]	navigate the menu. «Enter» selects submenus>. Highlighted letters are hotkeys. Pressing «Y» feature, while «N» will exclude a feature. Press «Esc»«Esc» to exit, «?» for Help, «/» for Search. feature is selected [] feature is excluded
	<pre>(rp100) System hostname (Welcome to Raspberry Pii) System banner Passwords encoding (md5)> /dev management (Bynamic using mdev)> Init system (Busybox)> (system/device table tx) Puth to the permission tables (NEW) Boot PS Seleton (default target skeleton)> () Root password (NEW) (ttyl) Port to run a getty (login prompt) on Baudrate Howse (189400)> (vt100) value to assign the TERM environment variable (NEW) [*] remount root filesystem read-write during boot (NEW) () Root filesystem overlay directories (NEM) () Root filesystem overlay directories (NEM) () Custom scripts to run before creating filesystem images (NEM)</pre>

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

row keys navigate the lectes a feature, whi gend: [*] feature is	nenu. <enter> selects suberus le <n> will exclude a feature. Pres selected [] feature is excluded</n></enter>	<pre>>. Highlighted letters are hotkeys. Pressing <y> is <esc><esc> to exit, <?> for Help, for Search</esc></esc></y></pre>
	<pre>cloop root filesystem for the target cpin the root filesystem (NEW) wt2/3/4 root filesystem (NEW) *** initumfs requires a Linux kerne iffs2 root filesystem (NEW) romfs root filesystem (NEW) romfs root filesystem (NEW) tar the root filesystem (NEW) Compression method (gzip)> other random options to pass to ta ubifs root filesystem (NEW)</pre>	device (NEW) an initial RAM filesystem) (NEW) il to be built *** br er (NEW)

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

rrow keys n electes a f egend: [*]	Narmal avigate the menu. <enter> selects submenus>. Highlighted letters are hotkeys. Pressing <t> sature, while <d> will exclude a feature. Press <esc> to exit, <t> for Help, for Search feature is selected [] feature is excluded</t></esc></d></t></enter>
	<pre>[*] Linux Kernel Kernel version (Custom Git tree)> (https://github.com/raspberrypi/Linux) URL of custom Git repository (rpi-3.9.y) Custom Git version () Custom kernel patches (MEW) Mernel configuration (Using a defconfig)> (bcorpi) Defconfig name Kernel bidary format (iinage)> [Device tree support (NEW) [] Install kernel image to /boot in targety(NEW) Linux Kernel Extensions></pre>

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 \sim 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:

GParted Ec	ev/sdd - GPa dit View D	rted evice Par	tition Help			-10
	-			[]/d	ev/sdd (7.4	0 GiB) 🛟
			/dev/sdd2 7.35 GiB	2		
Partition	File System	Label	Size	Used	Unused	Flags
/dev/sdd1	fat32	/boot	50.00 MiB	6.05 MiB	43.95 MiB	
/dev/sdd2	ext4	/rootfs	7.35 GiB	294.86 MiB	7.06 GiB	
0 operations	pending					

800 /	dev/sdd - GParted				
GParted E	Edit View Device	Partition Help			
	-1 E E	6 1		/dev/sdd	(7.40 GiB) 💲
		unallocat 7.40 GiB	ed		
Partition	File System	Size	Used	Unused	Flags
unallocate	ed 📃 unallocated	7.40 GiB		-	
0 operation:	s pending				

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:

Create new Partition		-			
Minimum size	e: 32 MiB	51	Maximum size	7.579 MiB	I.
Free space preceding (MiB):	1	•	Create as:	Primary Partition	12
New size (MiB):	50	:	File system	6233	1.
Free space following (MiB):	7529	12	The system.	CHARTE.	2.5
Align to:	MiB :		Label:	/boot b	
				Cancel Ad	ld.

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:

Create new Partition					U
Minimum s	ize: 1 Mit	1	Maximum size: i	7,529 MIB	ľ
Free space preceding (MIB):	0	1	Create as:	Primary Partition	3
Newsize (MiB):	7529	10	The successory	and the second sec	4
Free space following (MiB):	0		File system:	ext4	
Align to:	MiB	1	Label:	/rootfsj	
				Cancel Ad	d

🛞 🔿 💿 /dev/sdd	- GParted							
GParted Edit Vi	ew Device	Partition	Help			~		
0 -		€ √		R.	/dev/sdd	(7.40 GiB) 💲		
New Partition #2 7.35 GiB								
Partition	File System	Label	Size	Used	Unused	Flags		
New Partition #1	Fat32	/boot	50.00 MiB			-		
New Partition #2	ext4	/rootfs	7.35 GiB					
😼 Create Primary F	Partition #1 (fat32, 50.0	0 MiB) on /dev/sdd	ł				
🔓 Create Primary P	Partition #2 (ext4, 7.35	GiB) on /dev/sdd					
2 operations pendir	p							

Aplicar as modificações e fechar a janela que segue:

Applying pending operations		
Depending on the number and type of	of operations this might tak	e a long time.
Completed Operations:		
All operations su	iccessfully completed	
▶ Details		
	Save Details	Close

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 \sim -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 !"