Trabalho 1 - Sistemas Embarcados

André Luiz Granemann e Gabriela Alice Krieck

September 25, 2018

1 Objetivo

Este tutorial tem como objetivo implementar uma distribuição Linux que atue como um servidor Web. O servidor, por sua vez, fornecerá informações básicas sobre o funcionamento do sistema (*target*).

2 Premissas

Este tutorial é uma continuação dos tutoriais 1.1: Buildroot e QEMU e 1.2: Configurando a rede, disponibilizados no diretório "Tutoriais". Assim, para sua correta implementação, supõe-se que os tutoriais anteriores já foram implementados.

3 Configurando o buildroot

No diretório *buildroot/* da sua distribuição, entre na interface de configuração do Buildroot:

```
$ make menuconfig
```

Navegue nos menus, conforme mostrado abaixo e habilite a *toolchain* de suporte WCHAR:

```
Toolchain --->
    [*] Enable WCHAR support
```

Em seguida habilite o interpretador para o python e adicione os módulos externos conforme mostrado abaixo:

```
Target Packages --->
   Interpreter languages and scripting --->
       [*] python
       External python modules --->
           [*] python-psutil
```

Salve as configurações e saia do menu.

Certifique-se que o driver Ethernet está habilitado utilizando comando abaixo e as instruções do tutorial 1.1.

```
$ make linux-menuconfig
```

4 Criando o arquivo python

Acesse o diretório *buildroot/custom-scripts* e crie um arquivo *python* através do comando:

```
$ gedit <NOME-DO-ARQUVO>.py
```

Altere o *<NOME-DO-ARQUIVO.py>* para o nome desejado para seu arquivo python. Adicione ao arquivo o conteúdo abaixo:

```
import SimpleHTTPServer
import SocketServer
import os.path
import os, platform, subprocess, re
import datetime
import psutil

def get_processor_name():
    if platform.system() == "Windows":
        return platform.processor()
    elif platform.system() == "Darwin":
        os.environ['PATH'] = os.environ['PATH'] + os.pathsep + '/usr/sbin'
        command ="sysctl -n machdep.cpu.brand_string"
        return subprocess.check_output(command).strip()
    elif platform.system() == "Linux":
        command = "cat /proc/cpuinfo"
        all_info = subprocess.check_output(command, shell=True).strip()
        for line in all_info.split("\n"):
            if "model name" in line:
                return re.sub( ".*model name.*:", "", line,1)
    return ""

def get_cpu_usage():
    last_idle = last_total = 0
    with open('/proc/stat') as f:
        fields = [float(column) for column in f.readline().strip().split()[1:]]
    idle, total = fields[3], sum(fields)
    idle_delta, total_delta = idle - last_idle, total - last_total
    last_idle, last_total = idle, total
    utilisation = 100*(1-idle_delta / total_delta)
```

```
    return utilisation

def get_uptime():
     try:
         f = open( "/proc/uptime" )
         contents = f.read().split()
         f.close()
     except:
        return "Cannot open uptime file: /proc/uptime"

     total_seconds = float(contents[0])
     return total_seconds;

def getRAMinfo():

p = os.popen('free')
i = 0
while 1:
i = i + 1
line = p.readline()

if i==2:
memT=int(line.split()[1])/1000
memU=int(line.split()[2])/1000
mem = str(memU) + "Mb /" + str(memT) + "Mb"
return mem

def getPID():
list_proc = ""
for proc in psutil.process_iter():
try:
pinfo = proc.as_dict(attrs=['pid', 'name', 'username'])
except psutil.NoSuchProcess:
pass
else:
list_proc += "<p>" + str(pinfo) + "</p>"
return list_proc


#arquivo = open('index.html', 'w')
#arquivo.write(html)
#arquivo.close()


class MyRequestHandler(SimpleHTTPServer.SimpleHTTPRequestHandler):
    # Build the html file for every connection.
```

```
    def makeindex(self):
        tofile= """<!DOCTYPE html>
        <html>
        <body>

        <h1>Target do sistema</h1>

        <p> Uptime do sistema: %s segundos</p>

        <p> Nome do processador: %s </p>

        <p> Uso do processador: %s </p>

        <p> RAM usada / total: %s </p>

        <p> Versao do sistema: %s </p>

        <p>Relatorio gerado em: %s</p>

        <p> Lista de Processos em execucao:</p>
        %s

        </body>
        </html>
        """ % (str(get_uptime()), str(get_processor_name()), str(round(get_cpu_usage(), 2))
               str(platform.version()), str(datetime.datetime.now().strftime('%d/%m/%Y %H:%M
        f = open("index.html", "w")
        f.write(tofile)
        f.close()
        return

    # Method http GET.
    def do_GET(self):
        self.makeindex()
        self.path = '/index.html'
        return SimpleHTTPServer.SimpleHTTPRequestHandler.do_GET(self)


# Start the webserver.
Handler = MyRequestHandler
server = SocketServer.TCPServer(('0.0.0.0', 8080), Handler)
server.serve_forever()
```

Salve o arquivo e feche-o.

5 Configurando a inicialização do servidor

Acesse o diretório *buildroot/custom-scripts* e crie um arquivo chamado *S42server.sh* através do comando:

```
$ gedit S42server.sh
```

Adicione ao arquivo o conteúdo abaixo:

```
#!/bin/sh
python /usr/bin/<NOME-DO-ARQUIVO>.py &
```

Altere o *<NOME-DO-ARQUVO.py>* para o nome do seu arquivo python, salve o arquivo *S42server.sh* e feche-o. Ainda no diretório *buildroot/custom-scripts* edite o arquivo *pre-build.sh* utilizando o comando:

```
$ gedit pre-build.sh
```

O arquivo possuirá o conteúdo abaixo:

```
#!/bin/sh

  cp $BASE_DIR/../custom-scripts/S41network-config $BASE_DIR/target/etc/init.d
  chmod +x $BASE_DIR/target/etc/init.d/S41network-config
  cp $BASE_DIR/../custom-scripts/<NOME-DO-ARQUIVO.py> $BASE_DIR/target/usr/bin
  chmod +x $BASE_DIR/target/usr/bin/<NOME-DO-ARQUIVO.py>
  cp $BASE_DIR/../custom-scripts/S42server.sh $BASE_DIR/target/etc/init.d
  chmod +x $BASE_DIR/target/etc/init.d/S42server.sh
```

Altere o *<NOME-DO-ARQUVO.py>* para o nome do seu arquivo python, salve o arquivo *pre-build.sh* e feche-o.

6 Montando o sistema e inicializando

No diretório *buildroot/*, execute os comandos:

```
$ make clean
$ make
```

Para iniciar o sistema, execute a emulação do guest com o comando do QEMU, conforme abaixo:

```
sudo qemu-system-i386 --device e1000,netdev=eth0,mac=aa:bb:cc:dd:ee:ff \
   --netdev tap,id=eth0,script=custom-scripts/qemu-ifup \
   --kernel output/images/bzImage \
   --hda output/images/rootfs.ext2 \
   --nographic \
   --append "console=ttyS0 root=/dev/sda"
```

Para verificar o funcionamento do sistema, acesse pelo host em um navegador de sua preferência o link: `http://192.168.1.10:8080`. Para verificar o *status* atual do sistema basta atualizar a página.

Disponível em: `https://github.com/andregranemann/Trabalho_embarcados`