Universidade Federal do ABC

Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas

ESZB026-17 – Sistemas Embarcados para Engenharia Biomédica
2º quadrimestre de 2025

Roteiro 01:
Introdução ao Linux

1 Objetivo

Esta aula prática tem por objetivo apresentar algumas das ferramentas básicas do sistema operacional Linux, voltadas para o gerenciamento de sistemas embarcados, em particular sistemas que utilizam uma RaspberryPi. Vocês irão se ambientar com o sistema operacional Linux, realizando algumas operações básicas como:

1. Explorar a estrutura de diretórios do sistema operacional;
2. Aprender alguns comandos básicos no console;
3. Gerenciar os processos em execução;
4. Instalação de novos pacotes;
5. Conceitos de usuários, grupos e suas permissões.

2 Materiais

Neste roteiro você irá precisar de:

- 1 RaspberryPi 3 com fonte de alimentação, cabo hdmi, cabo de rede, teclado e mouse
- 1 computador rodando Windows ou Linux com acesso à rede local

3 Introdução

Uma distribuição Linux é uma versão Linux disponível publicamente que inclui um conjunto de programas e ferramentas. Existem muitas distribuições diferentes do Linux, cada qual focada em uma aplicação diferente. Cerca de 90% dos 500 supercomputadores mais poderosos do mundo usa o Linux.

Na base de todas as distribuições, está o núcleo Linux (ou *Linux kernel*), criado por Linus Torvalds, um estudante com 21 anos de idade, em 1991. O kernel do sistema é responsável por controlar as interações entre o hardware e outros programas da máquina. O kernel traduz as informações que recebe ao processador e aos demais elementos eletrônicos do computador.

4 Linux em sistemas embarcados

Para decidir qual distribuição Linux instalar em um sistema embarcado, algumas características da distribuição devem ser levadas em consideração:

- deve ser estável e bem suportada;
- deve possuir um bom gerenciador de pacotes;

- deve ser simples e ocupar pouco espaço de armazenamento;
- deve ter um bom suporte da comunidade para a sua plataforma;
- deve possuir *drivers* suportados para os periféricos que se pretende utilizar.

Existem diversas distribuições Linux para sistemas embarcados. Todas elas possuem o núcleo Linux, mas cada qual possui diferentes ferramentas e configurações que resultam em uma experiência de uso diferente. A versão oficial para as placas RaspberryPi é chamada de **Raspberry Pi OS**, e é baseada na distribuição Debian.

4.1 Configurando sua RaspberryPi

4.1.1 Definindo a senha do usuário `pi`

Para definir a senha do usuário `pi`, você deve seguir os seguintes passos:

- Abra um terminal (ícone com `>_` na barra superior)
- Digite o comando `passwd pi`
- Quando solicitado, altere a senha do usuário "pi" de "raspberry" para "ufabcESZB026-17"

Para o relatório

1) Coloque no seu relatório a saída dos comandos acima. Para isso, após digitar os comandos em um terminal, aperte a tecla "`Prt Scr`" de seu teclado. Ao fazer isso, aparecerá no seu diretório pessoal (`/home/pi/`) uma figura .png com uma cópia da tela. **Use essa tecla toda vez que quiser registrar o que aparece na tela do seu monitor**. Você pode, posteriormente, recortar apenas a região de interesse da imagem da tela com um editor de imagens qualquer.

4.1.2 Dando um nome para sua RaspberryPi

Por padrão, a RaspberryPi é configurada para, ao ser conectada a uma rede, requisitar um endereço IP DHCP (*Dynamic Host Configuration Protocol*). Esse serviço geralmente é fornecido por um servidor DHCP que está rodando na rede local. O servidor DHCP fornece endereços IP de maneira dinâmica a partir de uma lista de endereços. Após um certo tempo (definido na configuração do servidor DHCP), essa lista é redefinida e, ao conectar novamente à rede, RaspberryPi é alocada em um endereço IP diferente, o que dificulta o acesso remoto à mesma, fazendo com que seja necessário descobrir frequentemente qual endereço IP a RaspberryPi está utilizando.

Existem duas maneiras de evitar esse problema. A primeira é configurar a RaspberryPi com um IP estático, que nunca muda. Entretanto essa opção nem sempre está disponível na rede local. Outra maneira é definindo um nome para identificar sua RaspberryPi na rede. Por padrão, a RaspberryPi vem configurada com o nome `raspberrypi`. Porém, ao conectar mais de uma RaspberryPi em uma mesma rede, é necessário alterar esse nome para que seja possível acessá-las.

Para alterar o nome da sua RaspberryPi, é necessário editar dois arquivos. Primeiro, abra um terminal na RaspberryPi e edite o arquivo `/etc/hosts` com o comando:

```
sudo nano /etc/hosts
```

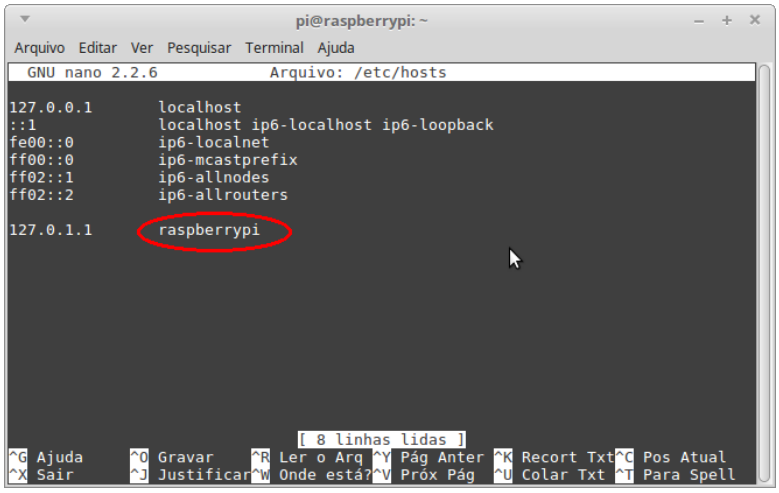

Figura 1: Abrindo o arquivo /etc/hosts com o programa `nano`.

Altere o nome da RaspberryPi ("raspberrypi") para "ESZB026-17-X", onde X é o número do seu grupo. Para navegar pelo arquivo, é necessário utilizar as setas do teclado. Pressione `Ctrl-X` para sair, confirmando a gravação das alterações.

Em seguida, no mesmo terminal, edite o arquivo `/etc/hostname` com o comando:

```
sudo nano /etc/hostname
```

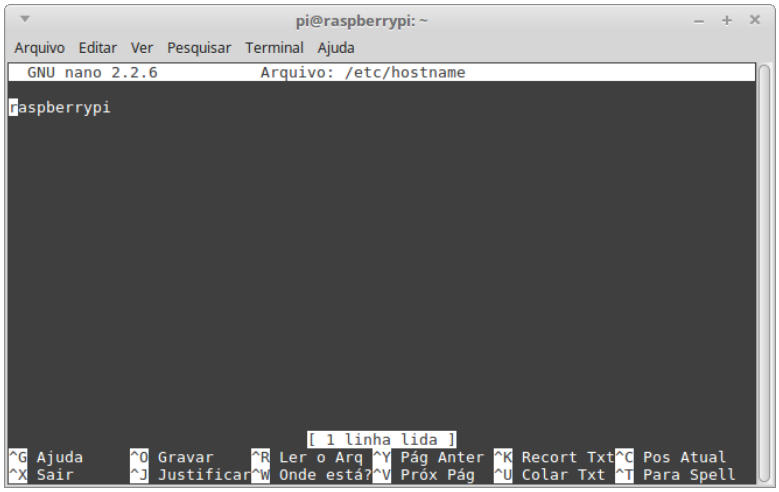

Figura 2: Abrindo o arquivo /etc/hostname com o programa `nano`.

Altere o nome da RaspberryPi ("raspberrypi") para "ESZB026-17-X", onde X é o número do seu grupo. Pressione `Ctrl-X` para sair, confirmando a gravação das alterações.

Confirme que as alterações foram gravadas conforme desejado, verificando o conteúdo dos dois arquivos. Para isso, use os seguintes comandos:

```
cat /etc/hosts
cat /etc/hostname
```

Para o relatório

2) Coloque no seu relatório a saída dos comandos `cat /etc/hosts` e `cat /etc/hostname`.

Após confirmar que os arquivos estão com o conteúdo correto, reinicie sua RaspberryPi com o comando:

```
sudo reboot
```

4.1.3 Testando a conexão com a RaspberryPi

Após a inicialização do sistema, teste a conexão, a partir de outro computador, com o comando `ping`. O comando `ping` é usado para testar a habilidade de um computador de se comunicar um outro computador na rede. Para isso, abra um terminal em outro computador e digite:

`ping -c 3 ESZB026-17-X.local` (em sistemas Linux); ou

`ping -n 3 ESZB026-17-X.local` (em sistemas Windows)

Caso a conexão não funcione, tente utilizar o número IP da placa RaspberryPi ao invés do nome, como mostram os comando a seguir (onde `XXX.XXX.XXX.XXX` é seu endereço IP. Note que com o endereço IP não deve ser usado o termo `.local`.

`ping -c 3 XXX.XXX.XXX.XXX` (em sistemas Linux); ou

`ping -n 3 XXX.XXX.XXX.XXX` (em sistemas Windows)

Para o relatório

3) Coloque no seu relatório a saída do comando `ping`, mostrando que a conexão foi bem sucedida. Não se esqueça de adicionar o termo "`.local`" ao final do nome do computador, caso a conexão seja pelo nome, para que o sistema saiba que esse nome pertence a um dispositivo na sua rede local.

4.2 Comunicando-se com sua RaspberryPi

Após configurar a rede na sua RaspberryPi e confirmar que ela é acessível por outros computadores na sua rede, é possível comunicar-se com sua RaspberryPi. Para isso iremos utilizar uma sessão *SSH* (*Secure Shell*). *SSH* é um protocolo de rede para comunicação segura e encriptada entre dispositivos de rede. Você pode utilizar um terminal de cliente *SSH* para conectar ao servidor *SSH* rodando na porta 22 da sua RaspberryPi. Com isso é possível:

- logar remotamente à sua RaspberryPi e executar comandos;
- transferir arquivos de e para a RaspberryPi usando o *Protocolo de Transferência de Arquivos SSH* (*SFTP*);
- direcionar conexões X11 (do ambiente gráfico), o que permite realizar computação virtual pela rede (*virtual network computing*).

4.2.1 Habilitando o *SSH* na RaspberryPi

Para habilitar as coneções *SSH* na RaspberryPi, siga os seguintes passos:

- Abra o menu "Preferences"→"Raspberry Pi Configuration"
- Vá para a aba "Interfaces"
- Na linha "SSH", escolha "Enable"
- Clique em "OK"
- Reinicie sua RPi

4.2.2 Conexões *SSH* usando PuTTY

PuTTY é um programa para Windows, gratuíto e de código aberto, usado como emulador de terminal, console serial e cliente *SSH* que você pode utilizar para se conectar à sua RaspberryPi pela rede. Ele pode ser obtido em https://www.putty.org/. Em um computador rodando Windows, entre na pagina de downloads do site, vá até a opção "*Alternative binary files*" e faça o download dos arquivos `putty.exe` e `pdftp.exe`.

Para configurá-lo, escolha *SSH* como tipo de conexão, entre com o endereço IP (ou *hostname*) da sua RaspberryPi, confirme a porta 22, e grave a sessão com um nome qualquer. Clique em *Open* e entre no sistema com seu nome de usuário (`pi`) e senha.

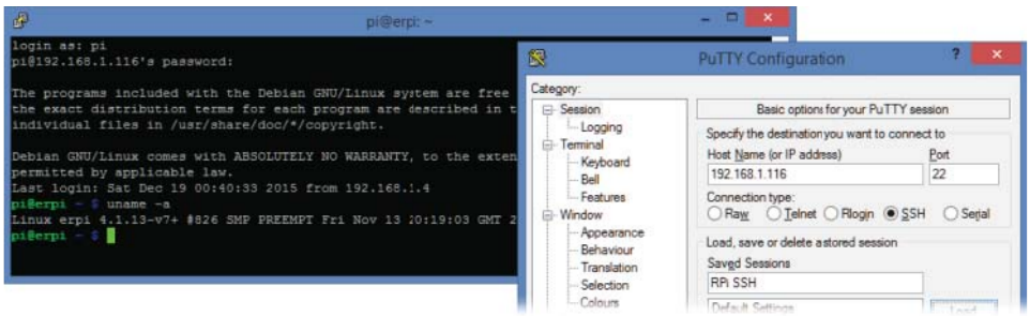

Figura 3: Ajustes de configuração do *SSH* no PuTTY ao lado de uma janela de terminal *SSH*.

Caso você esteja utilizando um computador com Linux, você pode utilizar o programa `ssh` ao invés do PuTTY.

4.2.3 Conexões *SSH* usando o Chrome Apps

O navegador de internet Chrome possui um cliente de *SSH* e emulador de terminal disponível. Vá ao *Chrome Web Store* e procure por "*Secure Shell*". Uma vez instalado, irá aparecer um ícone para o aplicativo *Secure Shell* quando você clicar no ícone de aplicativos do navegador. Ao iniciar esse aplicativo, configure a conexão conforme a figura 4, e o aplicativo irá aparecer conforme a figura 5.

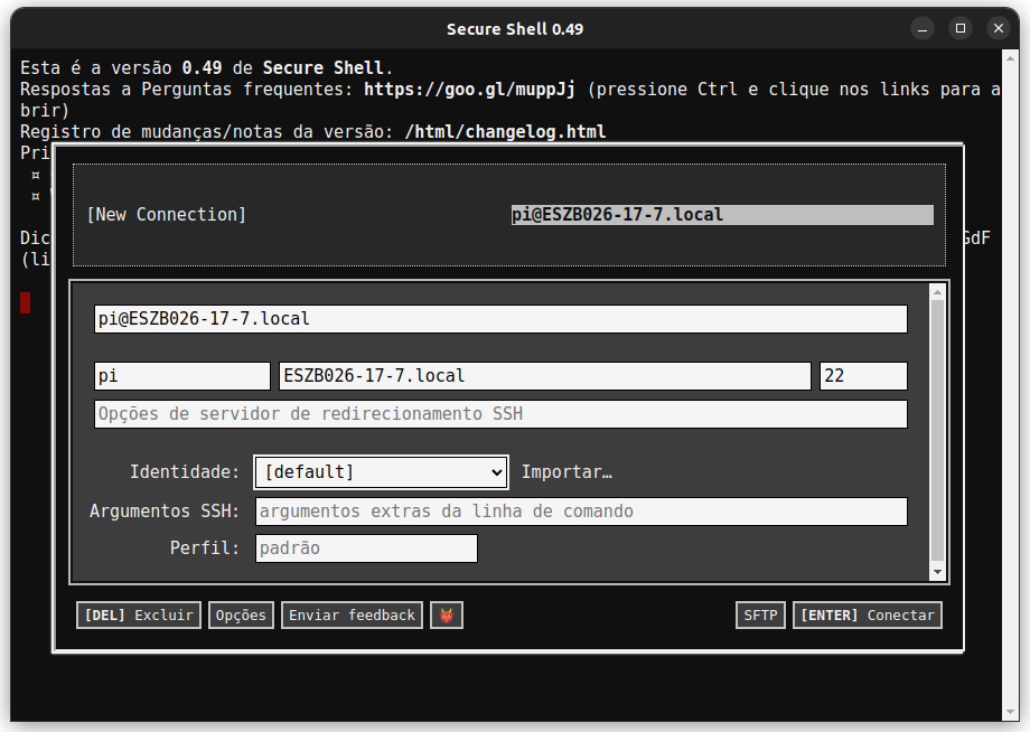

Figura 4: Tela de configuração da extensão "*Secure Shell*".

Figura 5: O aplicativo "*Secure Shell*" do Chrome.

Para o relatório

4) Conecte-se à sua RaspberryPi via *SSH* e rode os comandos `uname -a` e `date` na RaspberryPi a partir de outro computador. Inclua a saída obtida no seu relatório. Você pode utilizar o PuTTY, `ssh` ou Chrome para isso.

4.2.4 Copiando arquivos com PuTTY

O PuTTY inclui um programa para transferência de arquivos pela rede, chamado *psftp* (*PuTTY secure file transfer protocol*). Ao iniciar o *psftp*, digite o comando `open pi@ESZB026-17-X.local` (ou com o endereço IP da placa) para se conectar à RaspberryPi no endereço `ESZB026-17-X.local` com o usuário `pi`. Ao se conectar, você é automaticamente direcionado para o diretório `/home/pi/`. Use o comando `cd` para alterar o diretório atual. O comando `put nome_do_arquivo` envia um arquivo para a placa, e o comando `get nome_do_arquivo` transfere um arquivo da placa para seu computador local. Para transferir múltiplos arquivos, utilize os comandos `mput` e `mget`.

Comandos iniciados com o prefixo `l` (letra L minúscula) são executados no computador local (em algumas versões do *psftp* é utilizado o caracter '`!`' no lugar do '`l`').

Caso você esteja utilizando um computador com Linux, você pode utilizar o programa `sftp` ao invés do *psftp*.

4.3 Controlando sua RaspberryPi

Os comandos a seguir podem ser rodados diretamente na sua Raspberry Pi, em um terminal, ou através de uma conexão remota *SSH*. Em ambos os casos, você é direcionado a um prompt de comando. O caracter `$` no prompt indica que você está logado como um usuário comum. O caracter `#` no prompt indica que você está logado como super-usuário. Você deve evitar usar uma conta super-usuário sempre que possível, pois com essa conta é possível fazer alterações que podem danificar o seu sistema operacional.

A seguir são apresentados atalhos do teclado úteis e que funcionam na maioria dos sistemas Linux:

- ↑ (seta para cima): retorna o último comando digitado, e os comandos anteriores se pressionada novamente
- `Tab`: autocompleta o comando, nome do arquivo ou diretório que está sendo digitado. Se existirem muitas opções, pressionar `Tab` novamente irá mostrar uma lista de opções
- `Ctrl+A`: retorna ao início da linha sendo digitada
- `Ctrl+E`: retorna ao final da linha sendo digitada
- `Ctrl+U`: limpa a linha atual do início até a posição do cursor
- `Ctrl+L`: limpa a tela
- `Ctrl+C`: interrompe (mata) o processo sendo rodado

- `Ctrl+Z`: coloca o processo atual em segundo plano, parado. Ao digitar **`bg`**, o processo fica rodando em segundo plano. O comando **`fg`** traz o processo outra vez para o primeiro plano. O comando **`jobs`** mostra os processos rodando em segundo plano. O comando **`fg %N`**, onde N é o número de um processo, traz um processo específico para o primeiro plano

4.3.1 Comandos básicos do Linux

A seguir é apresentada uma lista com alguns comandos úteis do Linux.

- **`uname -a`** → mostra a versão do Linux que você está usando
- **`passwd`** → é usado para alterar a senha do seu usuário
- **`more /etc/issue`** → mostra a distribuição Linux sendo usada
- **`ps -p $$`** → mostra o *shell* sendo utilizado no momento (por exemplo, *bash*)
- **`whoami`** → mostra com qual usuário você está logado
- **`uptime`** → mostra a quanto tempo o sistema está rodando
- **`top`** → lista todos os processos e programas sendo executados. Para sair da lista, pressione `Ctrl+C`
- **`hostnamectl`** → mostra informações importantes sobre sua Raspberry Pi
- **`cal`** → mostra um calendário baseado em texto.
- **`cal 04 2018`** → mostra o calendário de Abril de 2018

Para o relatório

5) Inclua no relatório a saída dos comandos acima (com excessão do **`passwd`**).

4.3.2 Comandos básicos do sistema de arquivos

Os comandos a seguir podem ser utilizados para manipular um sistema de arquivos Linux. Algumas vezes pode ser necessário preceder o comando desejado pela palavra **`sudo`**, indicando que o comando deve ser rodado com privilégios de segurança de super-usuário. Como já dito anteriormente, muito cuidado ao usar comandos como super-usuário. O Linux assume que você sabe o que está fazendo e irá, por exemplo, apagar todos os arquivos do sistema operacional caso você peça para fazê-lo como super-usuário, sem reclamar.

- **`ls`**: lista os arquivos em um diretório. Argumentos úteis:

  `-a` mostra todos os arquivos, incluindo arquivos ocultos

  `-l` motra em formato de lista detalhada

  `-R` motra a listagem recursiva (incluindo subdiretórios)

  `-r` motra a listagem em ordem inversa

  `-t` ordena por data de modificação

  `-S` ordena por tamanho

  `-h` mostra tamanhos de arquivo de forma "legível"

  Exemplo: **`ls -alh`**

- **`pwd`**: mostra o diretório atual. Argumentos úteis:

  `-P` mostra o diretório físico, sem quaisquer links simbólicos

  Exemplo: **`pwd -P`**

- **`cd`**: muda o diretório atual. Argumentos úteis:

  `~` vai para o seu diretório pessoal (`/home/pi`)

  `..` vai ao diretório um nível acima

  Exemplo: **`cd /home/pi/lab01`** é o mesmo que **`cd ~/lab01`**

- **mkdir**: cria um diretório

  Exemplo: `mkdir teste`

- **rm**: apaga um arquivo ou diretório. Argumentos úteis:

  `-r` apaga de maneira recursiva. Usar para apagar diretórios com arquivos. Cuidado!
  `-d` apaga diretórios vazios
  Exemplo: `rm -r teste`
  Exemplo: `rm arquivo.txt`

- **cp**: copia um arquivo ou diretório. Argumentos úteis:

  `-r` copia de modo recursivo
  `-u` copia apenas se a origem for mais nova que o destino ou se não existe destino
  `-v` modo verbose, isto é, mostra o que está sendo copiado
  Exemplo: `cp a.txt b.txt`
  Exemplo: `cp -r teste teste2`

- **mv**: move um arquivo ou diretório. Também pode ser usado para renomear. Argumentos úteis:

  `-i` pergunta antes de sobreescrever
  Exemplo: `mv a.txt c.txt`
  Exemplo: `mv a.txt ~/teste/`

- **touch**: cria um arquivo vazio ou atualiza a data de modificação de um arquivo existente.

  Exemplo: `touch d.txt`

- **more**: visualiza o conteúdo de um arquivo. Você pode usar a tecla espaço para mudar de página.

  Exemplo: `more d.txt`

Para o relatório

6) A partir de um terminal, crie um diretório chamado lab01 em seu diretório pessoal (`/home/pi`), e mova as figuras criadas com o `Prt Scr` neste roteiro para esse diretório. Renomeie cada figura de modo que o nome descreva seu conteúdo. **Nunca use espaços em nomes de arquivos e diretórios. Isso dificulta sua manipulação por terminais. Ao invés disso, você pode usar underline ('_').** Não se esqueça de incluir a extensão do arquivo (`.png`). Depois, liste os arquivos nesse diretório de 3 maneiras diferentes (por exemplo, ordenando por data em lista detalhada, ordenando por tamanho etc), e inclua as saídas no seu relatório, descrevendo em cada caso como estão sendo listados. Ao alterar arquivos e diretórios em sua pasta pessoal, **nunca** use o prefixo **sudo**, evitando assim problemas maiores caso cometa um engano ou erro de digitação.

4.3.3 Variáveis de ambiente

Variáveis de ambiente são variáveis que descrevem a configuração do seu ambiente Linux, como o editor padrão ou a localização dos arquivos executáveis.

Para obter uma lista das variáveis de ambiente ajustadas na RaspberryPi, use o comando **env**.

Você pode ver o valor de uma variável específica com o comando **echo $nome_da_variável**. Por exemplo, para ver o conteúdo da variável `PATH`, que guarda os diretórios onde o sistema procura por um arquivo executável quando o usuário roda um comando, você deve usar o comando:

```
echo $PATH
```

Para alterar ou criar uma nova variável de ambiente, use o comando **export**. Por exemplo, na variável `PATH` os diretórios são separados por ':'. Para incluir um novo diretório (por exemplo, `/home/pi`), e manter os diretórios já definidos, use o comando:

```
export PATH=$PATH:/home/pi
```

Para o relatório

7) Inclua no relatório o conteúdo da variável `PATH`. Em seguida, inclua o diretório `/home/pi/lab01` na lista de diretórios do `PATH` e mostre novamente o conteúdo da variável `PATH`, confirmando a inclusão do diretório.

Essa alteração é perdida ao se desligar ou reiniciar a RaspberryPi. Para alterar variáveis ambiente de modo permanente em um shell bash, você deve modificar o arquivo `.profile`.

4.3.4 Gerenciamento de pacotes

Um gerenciador de pacotes é um conjunto de ferramentas que automatiza o processo de instalação, configuração, atualização e remoção de pacotes de software no sistema operacional Linux. Diferentes distribuições do Linux usam diferentes gerenciadores de pacotes. O Debian, o Ubuntu e o Raspbian usam o `APT` (*Advanced Packaging Tool*). Utilizando o pacote `nano` como exemplo, listamos a seguir os principais comandos de gerenciamento de pacotes:

- **`sudo apt install nano`** → instala o pacote `nano`
- **`sudo apt update`** → atualiza a lista de pacotes disponíveis
- **`sudo apt upgrade`** → atualiza os pacotes instalados no sistema
- **`dpkg-query -l | grep nano`** → verifica se o `nano` está instalado
- **`apt-cache search nano`** → verifica se existe um pacote contendo a palavra `nano` disponível
- **`apt-cache show nano`** → mostra informações sobre o pacote `nano`
- **`apt-cache policy nano`** → mostra outras informações sobre o pacote `nano`
- **`apt help`** → mostra a ajuda do comando `apt`
- **`apt-get download nano`** → faz o download do pacote `nano` no diretório atual
- **`sudo dpkg -i nome_do_pacote.deb`** → instala o pacote `nome_do_pacote.deb` previamente baixado
- **`sudo apt remove nano`** → remove o pacote `nano`
- **`sudo apt-get autoremove`** ou **`sudo apt-get clean`** → remove pacotes antigos não usados

A seguir vamos verificar como instalar um pacote qualquer (Wavemon, neste caso) de diferentes maneiras, e como removê-lo do sistema.

Para o relatório

8) O Wavemon é uma ferramenta útil para configurar conexões Wi-fi. A seguir iremos instalar esse pacote de diferentes maneiras, a partir de um terminal local ou via *SSH*. Para isso, rode a sequência de comandos a seguir (nessa ordem) e inclua as saídas no relatório.

- **`wavemon`** → Como esse pacote não vem instalado por padrão, o shell deve avisar que o comando não foi encontrado.
- **`sudo apt update`** → atualiza a lista de pacotes disponíveis (responda sim em caso de confirmação de atualizações).
- **`apt-cache search wavemon`** → Verifica se esse pacote existe.
- **`sudo apt install wavemon`** → Instala o pacote `wavemon`.
- **`wavemon`** → Agora, como o pacote está instalado, o programa deve mostrar informações sobre sua conexão Wi-fi.
- **`sudo apt remove wavemon`** → Remove o pacote `wavemon`.
- **`wavemon`** → Confirma que o pacote não está mais instalado. O shell deve avisar que o comando não foi encontrado.
- **`apt-get download wavemon`** → Baixa o pacote.
- **`ls -l wavem*`** → Lista os arquivos no diretório atual começados por `wavem`. O pacote baixado deve aparecer.
- **`sudo dpkg -i nome_do_pacote.deb`** (substituindo o nome do pacote pelo nome do arquivo baixado) → Instala o pacote baixado.
- **`wavemon`** → Como o pacote está instalado novamente, o programa deve mostrar informações sobre sua conexão Wi-fi.

4.4 Configurando sua RaspberryPi

As opções de configuração da RaspberryPi podem ser acessadas via interface gráfica, pelo menu "*Preferences*" → "Raspberry Pi Configuration". Uma janela irá abrir, permitindo alterar as configurações de sua placa. As mesmas opções de configuração também podem ser acessada em modo texto, por um terminal local ou *SSH*, através do comando:

```
sudo raspi-config
```

Lembre-se que, neste caso, a navegação pelos menus deve ser feita através do teclado. Esse comando será utilizado futuramente.

4.5 Desligando sua RaspberryPi

Nunca desligue sua RPi tirando o cabo de alimentação. O correto é desligá-la por *software*. Isto ocorre porque é preciso desmontar o cartão SD antes de desligar o sistema. Caso contrário, se o sistema for desligado durante uma operação de gravação na memória do cartão, o sistema de arquivos pode se corromper e o sistema pode parar de funcionar.

Para deligar o sistema, você deve escolher a opção na interface gráfica ou, no terminal, utilizar os comandos `shutdown` ou `reboot`:

- `sudo shutdown -h now` → desliga o sistema corretamente.
- `sudo shutdown -h +5` → desliga o sistema daqui a 5 minutos.
- `sudo reboot` → reinicia o sistema corretamente.

4.6 Gerenciando serviços

Um gerenciador de serviços e sistemas inicia e interrompe serviços (por exemplo, servidor Web, servidor *SSH* etc) dependendo do estado atual da RaspberryPi. O gerenciador de serviços e sistemas `systemd` é utilizado por padrão em sistemas Debian e Raspbian atuais.

Você pode utilizar o comando `systemctl` para inspecionar e controlar o estado do `systemd`. Você pode ver a lista de serviços que estão rodando na RaspberryPi rodando o comando sem argumentos:

```
systemctl
```

Como você pode perceber, existem diversos serviços rodando na sua placa. Para gravar a saída do comando acima em um arquivo texto, podemos fazer redirecionamento da saída com o símbolo `>`. Esse símbolo cria um novo arquivo (substitui o arquivo caso já exista), e grava nele a saída do comando executado antes dele. Caso você deseje anexar o conteúdo ao final do arquivo sem substituí-lo, pode usar os símbolos `>>`. Execute o comando:

```
systemctl > saida.txt
```

Em seguida, rode o comando `ls` e perceba que, agora, existe um arquivo `saida.txt` contendo a saída do comando `systemctl`.

Para o relatório

9) Inclua a saída do comando acima, com a lista dos processos rodando em sua placa, em seu relatório.

A seguir são apresentados alguns argumentos úteis do `systemctl`:

- `systemctl status nome_do_serviço` → verifica qual o estado do seviço `nome_do_serviço`.
- `systemctl start nome_do_serviço` → inicia um serviço
- `systemctl stop nome_do_serviço` → interrompe um serviço
- `systemctl enable nome_do_serviço` → habilita um serviço a iniciar durante o boot
- `systemctl disable nome_do_serviço` → desabilita a inicialização de um serviço durante o boot
- `systemctl is-enabled nome_do_serviço` → mostra se a inicialização de um serviço está habilitado durante o boot
- `systemctl restart nome_do_serviço` → reinicia um serviço
- `systemctl condrestart nome_do_serviço` → reinicia um serviço apenas se já estiver rodando
- `systemctl reload nome_do_serviço` → recarrega os arquivos de configuração de um serviço sem reiniciá-lo

4.6.1 Atualizando a data e hora da sua RaspberryPi

Você pode ter percebido que, ao ligar sua RaspberryPi, a hora ajustada no sistema pode estar errada, mesmo tendo sido ajustada anteriormente. Isso acontece porque a RaspberryPi não possui um relógio de tempo real ligado a uma bateria. Incluir essa funcionalidade adicionaria certa de $10,00 ao valor da placa, e foi propositalmente deixada de fora. Caso necessário, você pode ajustar manualmente a data e hora da sua placa com o comando `date`:

`sudo date -s "Sep 20 17:30"` altera a data para 20 de setembro, e a hora para 17:30.

Entretanto, exitem serviços do sistema operacional que automaticamente ajustam a data e hora da sua placa caso ela esteja conectada à internet. Um desses serviços é o `systemd-timesyncd` (que substituiu o `ntp`, usado nas versões anteriores do sistema operacional). Para verificar qual dos dois serviços de sincronismo de data e hora seu sistema usa, rode os comandos a seguir e analise a saída de cada um:

```
systemctl status systemd-timesyncd
systemctl status ntp
```

Para que o sincronismo de data e hora funcionem corretamente, é necessário que sua localização e fuso horário estejam configurados corretamente. Para verificar essa informação, use o comando:

```
timedatectl
```

Caso a informação esteja incorreta, configure novamente sua localização pelo comando `sudo raspi-config`.

Para o relatório

10) Verifique as informações de localização e fuso horário da sua placa, corrigindo-as caso necessário. Inclua no relatório a saída do comando `timedatectl`.

Os serviços `systemd-timesyncd` e `ntp` utilizam a porta 123 para se comunicar com um servidor de sincronismo de data e hora. Porém, em algumas redes, a porta 123 pode estar bloqueada por motivos de segurança. Nesses casos, mesmo com o serviço instalado, a hora não é atualizada. Você pode verificar que esse problema está ocorrendo observando a saída dos comandos acima: no caso do serviço estar instalado e mesmo assim não conseguir atualizar a hora, você verá uma mensagem de erro informando que o tempo de resposta do servidor foi esgotado. Para resolver esse problema, basta instalar o pacote `htpdate`, que faz o sincronismo utilizando a porta 80 (a mesma usada na navegação via http), verificando, em seguida, se o novo serviço consegue se conectar com o servidor. **Se sua RaspberryPi não estiver com a data correta**, instale e verifique o funcionamento deste pacote com os seguintes comandos:

```
sudo apt install htpdate
systemctl status htpdate
```

Para o relatório

11) Indique no relatório qual dos três serviços (`systemd-timesyncd`, `ntp` ou `htpdate`) está funcionando em sua placa. Em seguida, interrompa o serviço e reinicie o mesmo, verificando o estado do serviço a cada passo, rodando, nesta ordem, os comandos:

- `systemctl status nome_do_serviço`
- `sudo systemctl stop nome_do_serviço`
- `systemctl status nome_do_serviço`
- `sudo systemctl start nome_do_serviço`
- `systemctl status nome_do_serviço`

Inclua a saída desses comandos em seu relatório. Repare que, para iniciar e interromper serviços, você deve rodar o comando com privilégios de administrador.

5 *GitHub*

O *Git*, de maneira simplificada, é um sistema que permite rastrear alterações no conteúdo de um projeto de software conforme ele é desenvolvido. Ele foi desenvolvido por Linus Torvalds, e

é usado ainda hoje no desenvolvimento do núcleo do Linux. O *Git* é um sistema de controle de versões distribuído, ou seja, você mantém localmente uma cópia completa do repositório, incluindo seu histórico. Documentos alterados são marcados com números de revisão e hora de alteração. É possível comparar revisões e até reverter para versões antigas dos documentos.

A vantagem de um sistema distribuído é que você pode submeter (ou "*comitar*", do inglês *commit*) ao repositório alterações localmente, testá-las em seu sistema, sem a necessidade de alterar o repositório remoto, fazendo isso só quando tem certeza que a alteração funciona. Você pode criar desenvolvimentos paralelos (*branchs*) que não afetam o projeto principal.

Nesta disciplina iremos utilizar o *GitHub* como repositório remoto. Com a exceção de enviar alterações para o repositório remoto (*push*), você pode realizar todas as demais operações descritas aqui sem necessitar de uma conta. Como você pode perceber pela figura 6, quase todas as operações são realizadas localmente. Atualmente (2025) *GitHub* permite criar repositórios públicos e privados de graça, e cobra uma taxa para serviços adicionais, como manter um Wiki do seu projeto.

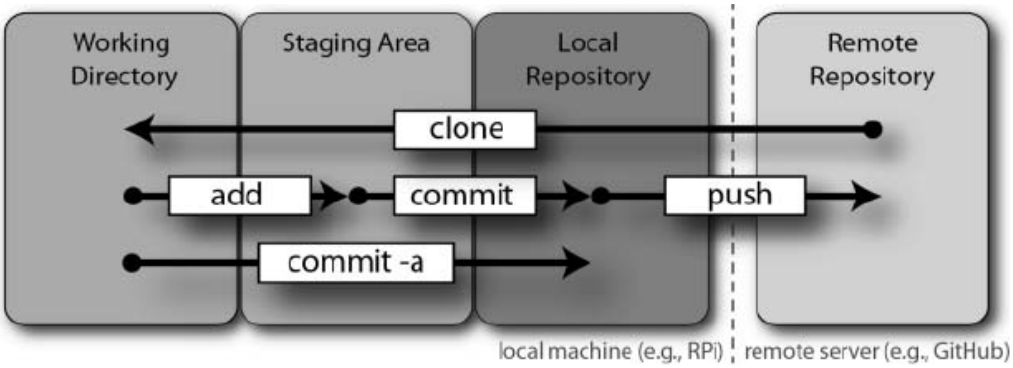

Figura 6: O fluxo de trabalho no *Git*.

5.1 Baixando um repositório existente (`git clone`)

Os códigos utilizados nesta disciplina estão disponíveis no *GitHub*, e podem ser acessados com o comando `git`. Caso o comando `git` não esteja instalado, você pode fazê-lo com o comando **`sudo apt install git`**. Você pode baixar os códigos rodando os seguintes comandos na RaspberryPi:

`cd ~` → navega para o seu diretório pessoal.

`git clone https://github.com/edlbcamargo/sist_embarcados_git` → faz uma cópia local do repositório remoto contendo os códigos desta disciplina.

`cd sist_embarcados_git` → navega para o repositório local.

`ls -la` → mostra os arquivos baixados.

Para o relatório

12) Inclua a saída dos comandos acima no seu relatório.

Futuramente, para verificar se novos códigos foram adicionados no repositório remoto (ou seja, no site do *GitHub*), você pode utilizar os comandos a seguir, de dentro do seu repositório local.

`git fetch` → atualiza as referências para o repositório remoto.

`git status` → mostra o estado do seu repositório local.

Caso seja identificado que seu repositório local está desatualizado, os novos arquivos podem ser baixados com o comando:

`git pull`

5.2 Criando um repositório

Você pode facilmente criar um repositório para esta disciplina. Caso ainda não tenha, crie uma conta gratuita no *GitHub* (https://github.com/). Cada membro do grupo deverá ter uma conta no *GitHub*, assim todos poderão incluir alterações, porém apenas um deverá criar o repositório. Em

seguida, escolha a opção "*New repository*". Entre com o nome do repositório: `ESZB026-17-X-2025B`, onde `X` é o número do seu grupo. Inclua uma descrição para o repositório, marque a opção para torná-lo disponível publicamente e a opção para inicializar com um arquivo `README`, e clique em `Create Repository`.

Crie agora as contas dos outros integrantes do grupo. Em seguida, inclua as outras contas como colaboradores, entrando na página do seu repositório no site do *GitHub*, navegando para `Settings`, marcando a opção `Restrict editing to collaborators only` e escolhendo a opção `Collaborators` no menu à esquerda da página.

Aviso!

O acesso ao *GitHub* via linha de comando por senha foi desabilitado por questões de segurança. Assim, para acessar seu repositório, cada usuário deverá criar um *token*. Saiba mais sobre como fazer isso em https://docs.github.com/pt/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token. Ao criar seu *token*, tome o cuidado de fornecer uma data de validade adequada, e marque as permissões `repo` e `admin:repo_hook`. Anote seu *token* com cuidado, pois você não conseguirá visualizá-lo novamente no futuro! Em 2025, o caminho para criar um *token* no *GitHub* era: "Settings" → "Developer Settings" → "Personal access tokens" → "Tokens (classic)" → "Generate new token" → "Generate new token (classic)".

Para o relatório

13) Inclua no seu relatório o link para seu repositório (algo como `https://github.com/edlbcamargo/sist_embarcados_git`), assim como os nomes de usuários de todos os integrantes do grupo. Neste repositório deverão estar todos os códigos criados pelo seu grupo, tanto para os roteiros quanto para o projeto.

5.3 Incluindo arquivos em seu repositório

Em seguida, crie uma cópia local do seu repositório com o comando `git clone`, dentro do diretório `/home/pi`. Para isso, antes de clonar o repositório, digite `cd ~`.

Navegue até seu repositório local e veja seu estado com o comando abaixo:

```
git status
```

Para que você não precise ficar digitando seu token a todo momento, vamos gravá-lo no sistema. Para isso, digite o comando abaixo[1]:

```
git config --global credential.helper store
```

Em seguida, digite `git pull` e entre com suas credenciais. Se o comando acima foi aplicado corretamente, o sistema irá gravar essas credenciais para usos futuros. Para verificar, digite `git pull` novamente, e dessa vez o sistema não deve pedir as credenciais de acesso.

Agora, dentro do seu repositório, crie duas pastas com o comando `mkdir`, uma chamada `lab01`, e outra chamada `Projeto`.

Para demonstrar a inclusão de um novo arquivo, vamos incluir um arquivo já existente. Para isso, copie o arquivo `lab01/exemplo.sh` do repositório da disciplina para a pasta `lab01` do seu repositório.

Veja novamente o estado do seu projeto, digitando `git status` outra vez. Você verá que o novo arquivo foi identificados, mas não é rastreado pelo seu repositório ainda. Para rastrear os novos arquivos (incluí-los no *Staging Area*), você deve usar o comando a seguir (não se esqueça do ponto no final):

```
git add .
```

Rode novamente o comando `git status`, e perceba que agora o arquivo é rastreado, mas ainda não foi enviado ao repositório local. Se futuramente você desejar remover um arquivo da lista de arquivos rastreados, você pode usar o comando `git rm arquivo.txt`.

[1]Se algum dia, por algum motivo, você precisar apagar as credenciais do sistema, isso é feito com o comando `git config --unset credential.helper`.

Antes de enviar (comitar) um arquivo para o repositório local, você deve configurar seu nome e usuário no `git`. Para isso, use os comandos a seguir. Essas informações não precisam, necessariamente, ser as mesmas informadas no site do *GitHub*.

```
git config --global user.name "Meu nome"
git config --global user.email meu@email.com
```

Você pode verificar essa configuração com o comando `more ~/.gitconfig`.

Para comitar os novos arquivos, isto é, enviá-los para o repositório local, você deve usar os comandos:

```
git commit -m "Incluindo o script bash de exemplo"
git status
```

Antes de enviar alterações do repositório local para o repositório remoto, você deve alterar uma configuração do `git` com o comando:

```
git config --global push.default simple
```

Uma vez configurado, novas alterações são enviadas para o repositório remoto com o comando a seguir (fornecendo na sequência seus dados do *GitHub*). Não se esqueça que você deverá usar seu *token* no lugar da senha para conseguir acesso.

```
git push
```

Para verificar as alterações em seu repositório, use o comando:

```
git log
```

Para o relatório

14) Inclua as saídas dos comandos acima no seu relatório.

Depois de enviar as alterações para o repositório remoto, você pode baixá-las em qualquer outro computador que já possuía seu repositório com o comando `git pull`.

A tabela a seguir mostra um resumo de outras operações comuns no `git`.

OPERATION	DESCRIPTION	OPERATION	DESCRIPTION
`git clone`	Clone from the remote repository.	`git rm`	Delete a file or directory from the staging area.
`git init`	Create a wholly new repository.	`git mv`	Move or rename a file or folder in the staging area.
`git pull`	Merge changes from a master repository.	`git log`	Display a log of commits. The project history.
`git fetch`	Find what has changed in a master repository without merging.	`git tag`	Give a commit a name (e.g., version 2).
`git status`	Show the project's status.	`git merge [name]`	Merge the branch.
`git add`	Add a new file or edit an existing file.	`git show`	Get details about the current or other commit.
`git diff`	Show the differences that are to be committed.	`git branch [name]`	Create a new branch. (Use `-d` to delete.)
`git commit`	Commit to the repository.	`git checkout [name]`	Switch to a different branch.
`git push`	Push changes from the local repository to a remote repository.		

Figura 7: Resumo das principais operações no `git`.

6 Projeto

Nesta disciplina seu grupo irá desenvolver um equipamento médico que utilize sistemas embarcados.

O projeto deve, obrigatoriamente:

- Utilizar um conjunto RaspberryPi + Arduino
- Medir e/ou alterar uma variável física
- Gerar notificações através de alarmes (sonoro e visual)
- Gerar um relatório via Web

Como exemplo de projeto, podemos desenvolver um termômetro, que notifica quando a temperatura sai de uma faixa pré-estabelecida, guarda e exibe um relatório via Web com o valor da temperatura ao longo do tempo, e possui uma saída analógica para alterar a potência de um lençol térmico. Outros exemplos de projetos são: análise de postura; monitoração da respiração; reabilitação; bomba de infusão; monitoração de pulso etc.

Para o relatório

15) Inclua em seu relatório uma descrição detalhada do projeto que seu grupo irá desenvolver, incluindo os sinais das prováveis entradas e saídas, interfaces com o usuário, informações que irão constar no relatório via Web, estados que levam a uma notificação etc.

Referências Bibliográficas

- MOLLOY D., Exploring Raspberry Pi: Interfacing to the real world with embedded linux, Wiley,2016.
- The Raspberry Pi Foundation Website (https://www.raspberrypi.org/). Acesso em Setembro de 2018.