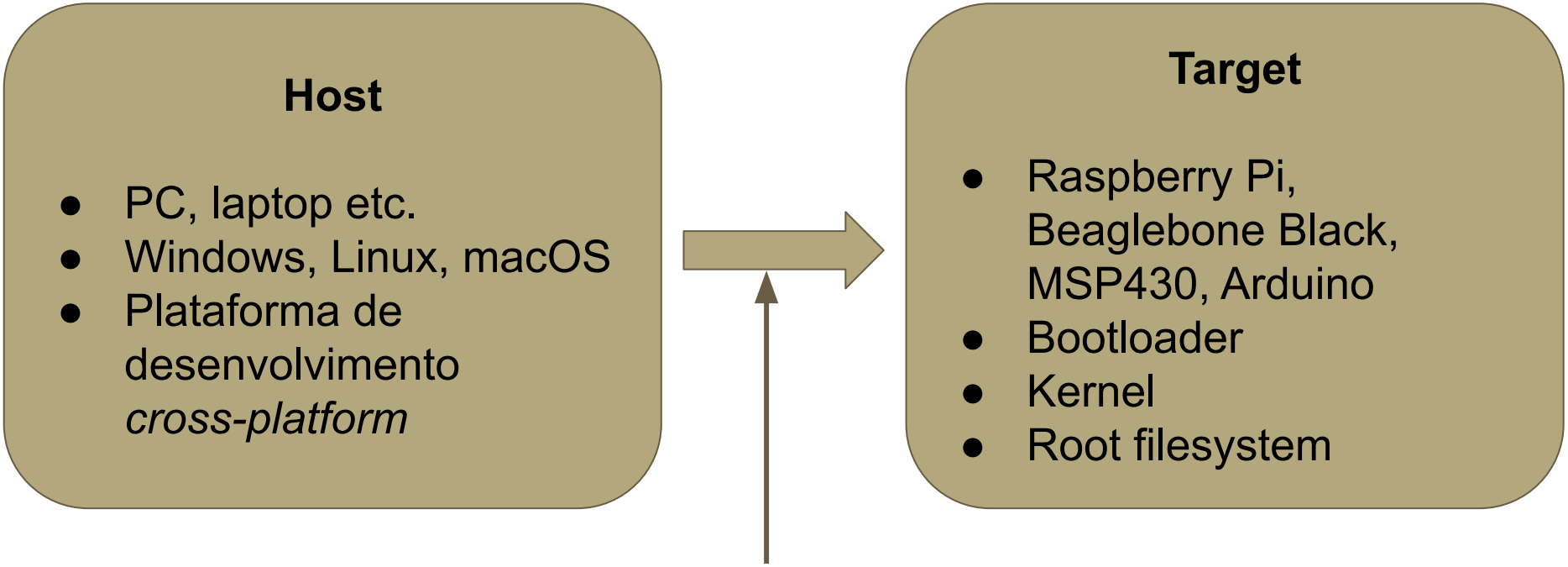
Sistemas Operacionais Embarcados

Desenvolvimento para sistemas embarcados

Desenvolvimento para sistemas embarcados

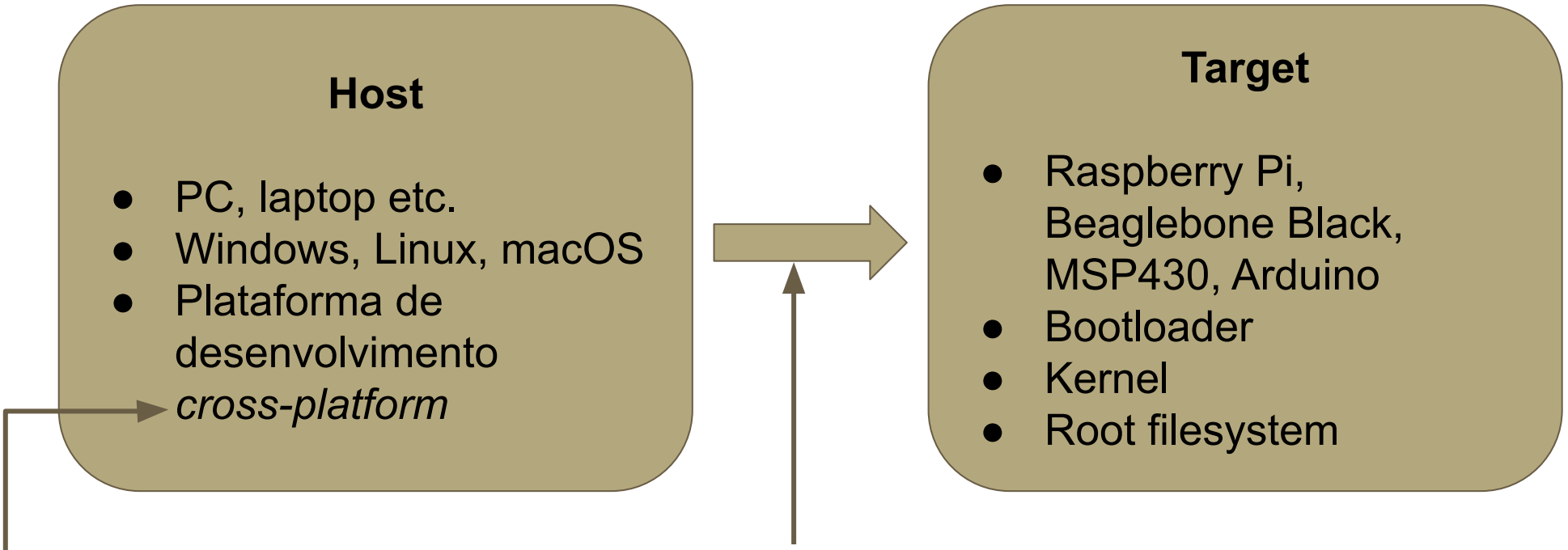
- Componentes e funções
  - Host
  - Target
  - Interface de comunicação
- Desenvolvimento *cross-platform*
- Controle de versão usando `git`

Componentes e funções

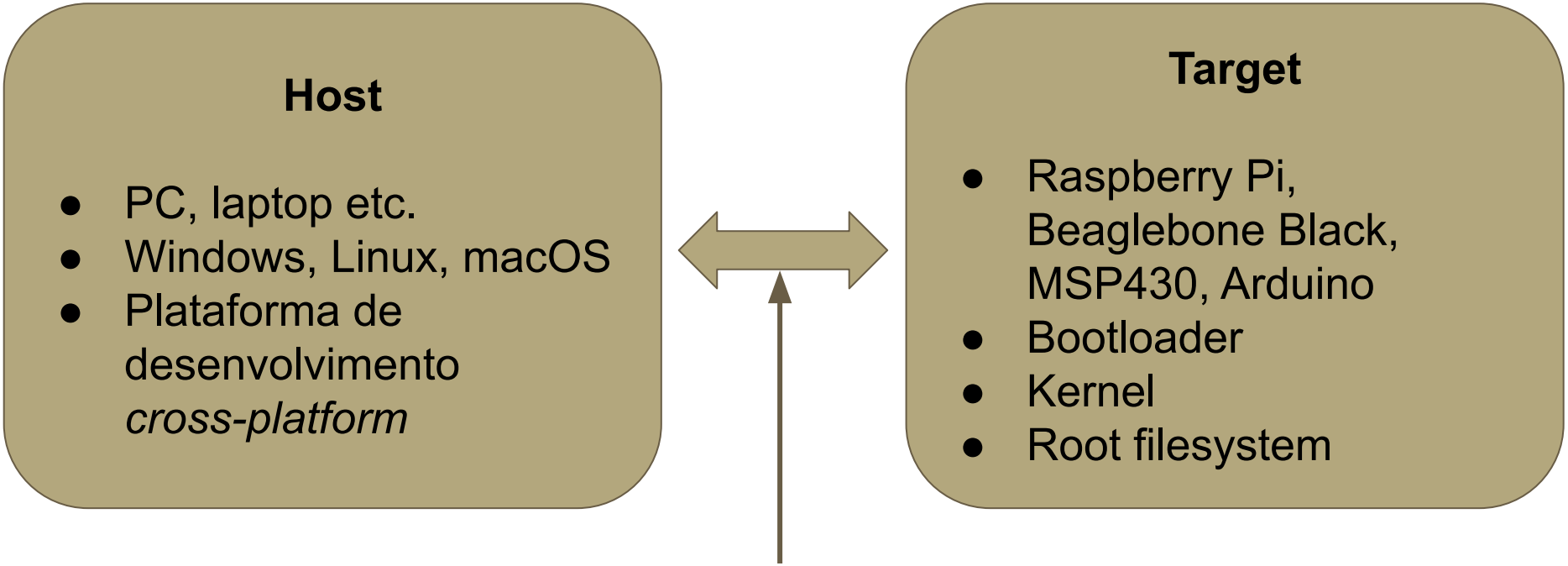


Interface de comunicação: cabo físico, WiFi etc.

Componentes e funções



Componentes e funções



Interface de comunicação: mídia removível (cartão SD, pendrive etc.)

Componentes e funções

Target

- Raspberry Pi, Beaglebone Black, MSP430, Arduino
- Bootloader
- Kernel
- Root filesystem
- Ambiente de desenvolvimento nativo (GCC etc.)

Componentes e funções

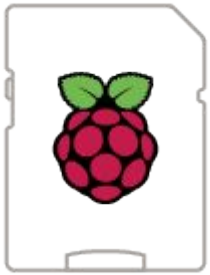
Target

- Raspberry Pi, Beaglebone Black, MSP430, Arduino
- Bootloader
- Kernel
- Root filesystem
- Ambiente de desenvolvimento nativo (GCC etc.)

Tudo é feito no *target*, que provavelmente terá menor poder computacional e menos memória do que um *host* típico

Sistema operacional para o RPi

Será necessário instalar em um cartão SD um sistema operacional para ser usado no Raspberry Pi:



- Raspberry Pi OS
  - Antigo Raspbian
  - SO oficial da RPi Foundation
  - Muita documentação
  - Baseado em Debian
  - Oferece versões com desktop e sem (somente terminal)
- Windows for IoT
- Ubuntu para RPi
- RetroPie (distribuição Linux para emular jogos retrô)
- OSMC (distribuição Linux para *media centers*)
- Kali Linux (distribuição Linux para testes de segurança de rede)
- Outros

Sistema operacional para o RPi

Antes de comprar um cartão SD, confira os requisitos necessários

Existem diversas formas de instalar o Raspberry Pi OS:

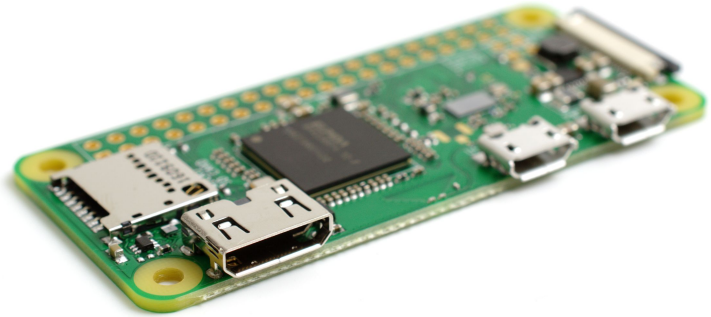
- Usando o Raspberry Pi Imager
- Diretamente pela linha de comando no Linux
- Diretamente pela linha de comando no macOS
- Usando programas específicos para Windows

Cenário 1: *target* independente

Target

- Raspberry Pi com sistema operacional Raspbian instalado
- Portas USB suficientes para conectar teclado e mouse
- Cabo HDMI + tela ou TV

1. Confira a necessidade de uma fonte adequada para o seu modelo
2. Confira a quantidade de portas USB disponíveis para o mouse e o teclado: RPi0 e RPi0w possuem somente uma, requerendo um hub de portas USB
3. Confira a necessidade de um conversor VGA/HDMI para a sua tela
4. https://projects.raspberrypi.org/pt-BR/projects/raspberry-pi-using/



Cenário 2: *host* + interface de comunicação

Host

- PC ou laptop
- Windows, Linux, macOS
- Plataforma de desenvolvimento *cross-platform*

Target

- Raspberry Pi com sistema operacional Raspbian instalado

Comunicação via cabo Ethernet

- Confira se o seu modelo possui porta Ethernet, ou consiga um adaptador Ethernet/USB
- Compartilhe a conexão à internet do seu computador com o Raspberry Pi via Ethernet
- Ative no RPi o VNC ou o SSH
- Descubra o IP do Raspberry Pi usando o Adafruit RPi Finder ou o comando `arp`
- Conecte ao RPi via VNC ou SSH

Cenário 2: *host* + interface de comunicação

Host

- PC ou laptop
- Windows, Linux, macOS
- Plataforma de desenvolvimento *cross-platform*

Target

- Raspberry Pi com sistema operacional Raspbian instalado

Comunicação v

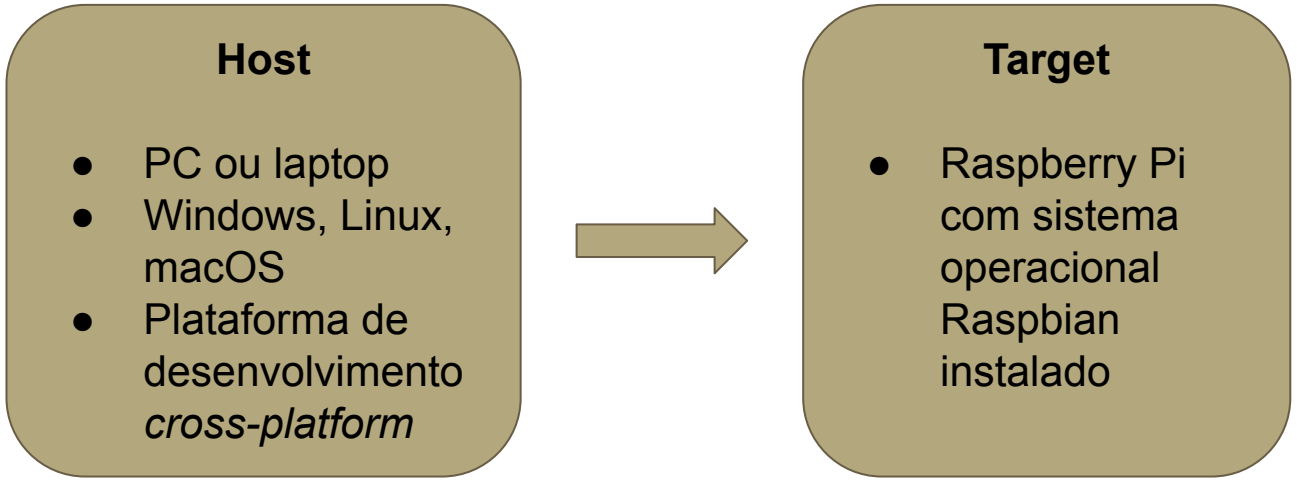
- Confira se
  consiga ur
- Compartilh
  com o Ras
- Ative no RPi o VNC ou o SSH
- Descubra o IP do Raspberry Pi usando o Adafruit RPi Finder ou o comando `arp`
- Conecte ao RPi via VNC ou SSH

```
$ ssh <usuario>@<endereço ip>

p.ex.

$ ssh root@192.168.7.2
```

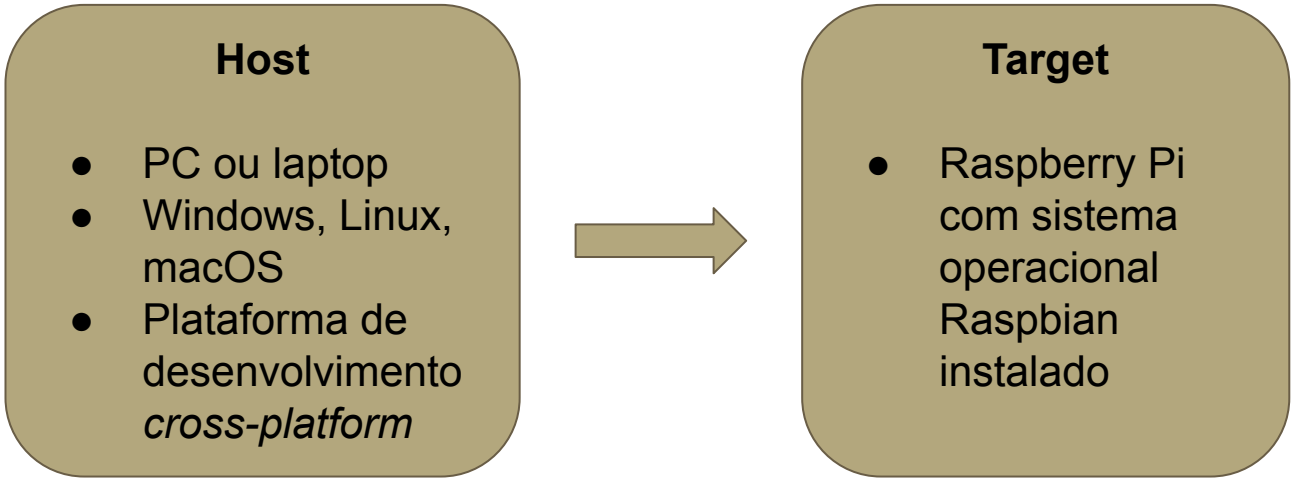
Cenário 2: *host* + interface de comunicação



Comunicação via WiFi

- Confira se o seu modelo possui WiFi
- Confira se sua rede permite a conexão entre computadores via WiFi
  - Redes caseiras geralmente não causam problemas
  - A rede WiFi da UnB não permite
- Conecte o RPi à rede WiFi, acessando-o diretamente com teclado, mouse e tela (etapa necessária somente no primeiro acesso)
- Ative no RPi o VNC ou o SSH
- Descubra o IP do Raspberry Pi usando o Adafruit RPi Finder ou o comando `arp`
- Conecte ao RPi via VNC ou SSH

Cenário 2: *host* + interface de comunicação



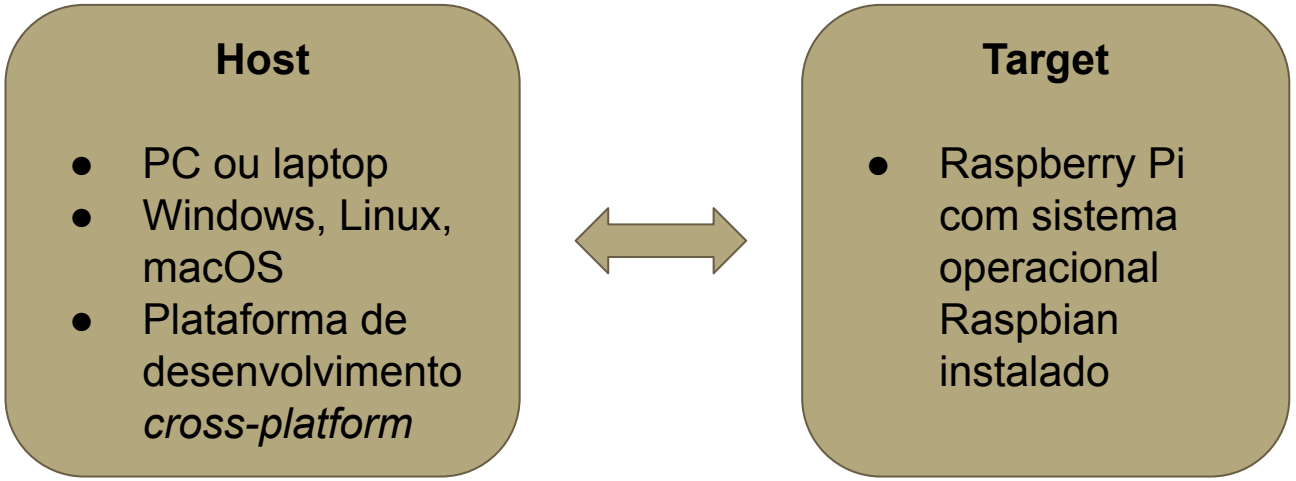
Comunicação via porta USB

- Confira se o seu modelo permite acesso remoto via porta USB
- Siga estas instruções ou estas instruções

Comunicação via porta serial

- Compre o cabo de conexão serial
- Siga estas instruções

Cenário 2: *host* + interface de comunicação



Comunicação via cartão SD

- Através do seu *host*, faça os mudanças no cartão SD (compilação de código etc.)
- Reinicie o RPi com o cartão atualizado
- Repita esses passos até terminar o projeto

Cross-compilation

- No cenário 2 (*host* + interface de comunicação), é possível que o *host* e o *target* tenham CPUs de diferentes arquiteturas
  - P. ex., PCs e laptops (Intel) e Raspberry Pi (ARM)

- O código compilado no *host* não funcionará no *target*, pois as instruções em Assembly são diferentes

- O *cross-compiler* permite compilar código para o *target* no *host*

- A RPi Foundation fornece gratuitamente um *cross-compiler*

- Depois de compilar seu código no *host*, é possível transferir o executável para o *target* usando o comando `scp`

Cross-compilation

- No cenário 2 (*host* + interface de comunicação), é possível que o *host* e o *target* tenham CPUs de diferentes arquiteturas
  - P. ex., PCs e laptops (Intel) e Raspberry Pi (ARM)

```
$ scp <caminho local> <usuario>@<ip>:<caminho remoto>

p.ex.

$ scp codigo.out aluno@192.168.7.2:/home/aluno
```

- A RPi Foundation fornece gratuitamente um *cross-compiler*
- Depois de compilar seu código no *host*, é possível transferir o executável para o *target* usando o comando `scp`

Controle de versão usando o `git`

- Trabalhar com código-fonte apenas localmente pode ser arriscado
  - Problemas com sistemas operacionais: atualizações com bugs, versões incompatíveis
  - Problemas com memória: HD, SSD
  - Outros problemas com hardware: fontes de alimentação, telas quebradas

- Uma solução simples é utilizar algum sistema de sincronismo de pastas: Dropbox, Google Drive, OneDrive etc.
  - Ferramentas típicas para escritório, mas insuficientes para desenvolvimento de software

- Uma solução amplamente utilizada é o `git`

Controle de versão usando o `git`

- Instale o git no *host*:
  ```
  $ apt-get install git
  ```
- Crie um repositório no GitHub, no GitLab, no Bitbucket ou no seu sistema de preferência
- Baixe seu repositório no seu *host*:
  ```
  $ git clone usuário@servidor:/caminho/para/o/repositório
  ```
- Será criada no seu *host* uma pasta com o nome do repositório

Controle de versão usando o `git`

- Acrescente arquivos de acordo com a necessidade do seu projeto
  - Código-fonte em C e C++
  - Scripts
  - Documentação (abuse do Markdown)
  - Sub-pastas

- O git não atualiza automaticamente. Para atualizar o repositório com as mudanças, vá para a raiz do repositório e digite

```
$ git add *
$ git commit -m “Comentários das alterações”
$ git push origin master
```

- No seu *target*, copie o repositório atualizado digitando

```
$ git clone usuário@servidor:/caminho/para/o/repositório
```

Controle de versão usando o `git`

- Mesmo que você não esteja usando um *host* separado (cenário 1), utilize o git para evitar perdas no projeto
  - Se o repositório estiver atualizado, é possível baixa-lo novamente
  - É possível retomar o repositório a partir de uma versão anterior, tomando por base os comentários das alterações
  - Além disso, é possível adicionar rótulos (*tags*) para indicar *releases* de *software*

- O git permite o uso de ramificações no projeto (*branches*), para que você teste variações e alternativas no seu projeto antes de efetiva-las