Ferramental

Vamos instalar o ferramental (compiladores) que será utilizado para compilar o kernel e o filesystem. Deveremos instalar o `socdes` e o `linaro-gcc` .

Intel SOCDES

Você irá precisar ter o software instalado (mesmas versões!): - Quartus - Intel SoC FPGA Embedded Development Suite

Vamos precisar inserir no path do bash referência para uma série de softwares a serem usados, modifique seu `.bashrc` inserindo:

Ao fazer o download do `soced` deve-se verificar para ser a mesma versão do Quartus.

```
export ALTERAPATH=~/intelFPGA/18.1/
export PATH=$PATH:${ALTERAPATH}/quartus/bin
export PATH=$PATH:${ALTERAPATH}/modelsim_ase/linuxaloem/
export PATH=$PATH:${ALTERAPATH}/quartus/sopc_builder/bin/
export PATH=$PATH:${ALTERAPATH}/embedded/host_tools/altera/
preloadergen/
export PATH=$PATH:${ALTERAPATH}/quartus/embedded/
export PATH=$PATH:${ALTERAPATH}/nios2eds/
export SOCEDS_HWLIB=${ALTERAPATH}/embedded/ip/altera/hps/
altera_hps/hwlib/
```

Note

Lembre de verificar se o **ALTERAPATH** desse exemplo é o caminho correto da instalação do Quartus

> **outros bashs**
>
> Se estiver usando outro bash (zsh/ fish) será necessário editar o arquivo de configuração referente.

Testando

Para testar, digite no terminal (após abrir uma nova aba, ou executar `source ~/.bashrc`):

```
$ nios2_command_shell.sh
-----------------------------------------------
Altera Nios2 Command Shell [GCC 4]

Version 16.1, Build 196
-----------------------------------------------

$ exit
```

> **Note**
>
> Isso só testa uma parte da instalação (soceds)

GCC toolchain

Iremos utilizar o GCC cross compile fornecido pelo Linaro, esse mesmo GCC será utilizado para compilar o Kernel, gerar o file system e compilar os programas que executarão no Linux. Para facilitar a organização, iremos manter o toolchain na pasta `~/work/`.

```
$ mkdir ~/work/
$ cd ~/work
$ wget https://releases.linaro.org/components/toolchain/binaries/
7.3-2018.05/arm-linux-gnueabihf/gcc-linaro-7.3.1-2018.05-i686_arm-
linux-gnueabihf.tar.xz
$ tar xvf gcc-linaro-7.3.1-2018.05-i686_arm-linux-gnueabihf.tar.xz
```

De uma olhada na pasta recém extraída:

```
$ cd gcc-linaro-7.3.1-2018.05-i686_arm-linux-gnueabihf
$ tree -L 1
...
+ arm-linux-gnueabihf
+ bin
 + arm-linux-gnueabihf-addr2line
 + arm-linux-gnueabihf-ar
 + ...
 + arm-linux-gnueabihf-c++
 + arm-linux-gnueabihf-g++
 + arm-linux-gnueabihf-gcc
+ include
+ lib
+ libexec
+ share
```

Temos todas as ferramentas necessárias para compilar e linkar códigos em C e C++ para o ARM.

Note que no path do gcc temos o prefixo : **gnueabihf**.

> **Pesquisa**
>
> Qual a diferença entre `eabi` e `hf`

Criando um atalho no bash

Vamos criar um atalho para essa pasta no bash. Edite o arquivo `~/.bashrc` para incluir a pasta `~/work/gcc-linaro.../bin/` na variável do sistema: **GCC_Linaro**.

```
# GCC Linaro on path
$ export GCC_Linaro=/home/corsi/work/gcc-linaro-7.2.1-2017.11-
x86_64_arm-linux-gnueabihf/bin
```

> **Note**
>
> Edite o comando para a pasta correta de onde Linaro foi extraído: `/home/...`

Agora temos um atalho para o gcc-arm, vamos testar :

```
$ $GCC_Linaro/arm-linux-gnueabihf-gcc -v
...
gcc version 7.2.1 20171011 (Linaro GCC 7.2-2017.11)
```