Judson Santos Santiago

Ambiente de Trabalho

Compiladores

Introdução

- **Linux** é um sistema operacional gratuito e de código aberto
  - Criado em 1991 por um estudante de Ciência da Computação da Universidade de Helsinki (Finlândia) chamado Linus Torvalds

> Hello everybody out there using **minix** –
>
> I'm doing a (free) operating system (just a hobby, won't be big and professional like **gnu**) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).
>
> I've currently ported **bash**(1.08) and **gcc**(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)
>
> Linus Torvalds

Introdução

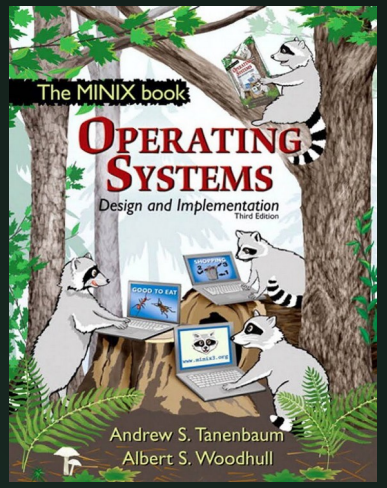
Linus Torvalds

Criador do Linux

Histórico

- Foi baseado no MINIX
  - Um sistema operacional desenvolvido por Andrew S. Tanenbaum em 1987 para ilustrar os princípios abordados em seu livro:

Operating Systems
Design and Implementation



3ª Edição

Andrew S. Tanenbaum

Histórico

- Linux utiliza software do GNU Project
  - GNU significa GNU is not Unix
  - Projeto criado por Richard Stallman em 1983 com o objetivo de criar um sistema operacional gratuito baseado no UNIX
    - O projeto GNU é responsável por muitos softwares utilizados no Linux:
      - **GCC**, **GDB** e **Make**: ferramentas de programação
      - **Emacs**: editor de textos
      - **Gzip**: compactador de arquivos
      - **Gnome**: gerenciador de janelas
      - Etc.

Richard Stallman

Histórico

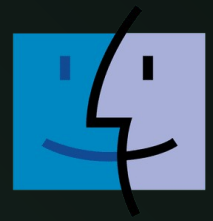
- O sistema operacional UNIX foi concebido e implementado no Bell Labs (AT&T) por Ken Thompson e Dennis Ritchie em 1969
  - Ken Thompson é o criador da primeira versão do **UNIX**
  - Dennis Ritchie é o criador da **linguagem C**

Ken Thompson

Dennis Ritchie

Histórico

- No fim dos anos 70, a AT&T licenciou o UNIX para universidades e empresas, dando origem a muitos sistemas “UNIX”:



  - **BSD** (Berkeley)
  - **Xenix** (Microsoft)
  - **AIX** (IBM)
  - **Solaris** (Sun Microsystems)
  - A versão mais popular de UNIX hoje é o **MacOS** (Apple)

Histórico

- Hoje, o que chamamos de **Linux** é:
  - O Kernel do Linux
    criado e mantido por Linus Torvalds
  - Muitos softwares do Projeto GNU
    iniciado por Richard Stallman
  - Software livre
    mantido pela comunidade (vários autores)
  - Extras
    drivers, software não-livre

A GNU recomenda chamá-lo de **GNU/Linux**

Distribuição

- **Software Livre** significa que ele pode ser executado, estudado, adaptado e redistribuído livremente
  - O código é aberto
  - Não necessariamente ele é gratuito
  - A GNU inicialmente vendia “pacotes de software”
- A maior parte do sistema é composto de software livre
  - Qualquer um pode baixar, empacotar e distribuir
  - Muitas empresas fazem suas próprias “distribuições” do Linux

Distribuição

- As distribuições mais populares:




Instalação do Linux

- Opções de instalação (sem remover o Windows):
  - Dual Boot
    - Recomendado: melhor experiência e desempenho
    - GRUB: gerenciador de boot
  - Máquina Virtual
    - Abstrai o hardware da máquina (VMware Player, VirtualBox)
    - Indicado quando se tem hardware não suportado
  - Bash no Windows Susbsystem for Linux (Windows 10)
    - Fornece apenas terminal e ferramentas em modo texto
    - Um servidor X pode ser usado para alguns aplicativos gráficos (VcXsrv)

Interface Gráfica

- Ao contrário do Windows, o Linux possui várias interfaces gráficas disponíveis:
  - **Gnome**, Cinnamon, MATE, Unity, **KDE**, Xfce, Etc.
  - Muitas são derivações de outras
- Algumas vem com um conjunto de aplicativos exclusivos:
  - Gerenciador de arquivos
  - Instalador de programas
  - Tocador de áudio
  - Etc.

Shell *versus* Terminal

- O Shell é um programa que permite ao usuário interagir com o S.O. através de comandos em modo texto
  - Antigamente era a única interface disponível em sistemas UNIX
  - O **Bash** (Bourne Again Shell) é utilizado na maioria das distribuições
    - Uma versão melhorada do Shell do UNIX, sh, criado por Steve Bourne
    - Existem outros disponíveis: sh, dash, csh, ksh, zsh, etc.
- Terminal é um programa que roda o Shell em uma janela
  - Existem vários: gnome-terminal, konsole, xterm, etc.

Terminal do Linux

- Alguns comandos básicos do Shell

Comando	Significado	Descrição
`ls`	`list`	Lista arquivos
`mkdir`	`make directory`	Cria diretório (pasta)
`rmdir`	`remove directory`	Remove diretório
`cd`	`change directory`	Muda de diretório
`cp`	`copy`	Copia arquivos
`mv`	`move`	Move arquivos
`rm`	`remove`	Remove arquivos

As opções de cada comando podem ser obtidas com `--help`

Terminal do Linux

- É comum a manipulação de arquivos pelo próprio terminal

Comando	Significado	Descrição
`cat`	`concatenate`	Concatena arquivos na saída padrão
`more`	`more`	Exibição controlada de arquivos
`less`	`less`	Navegação controlada em arquivos
`touch`	`touch`	Atualiza ou cria arquivo com a data atual
`>`	`redirect ouput`	Redireciona saída para arquivo
`>>`	`append ouput`	Adiciona saída em arquivo
`<`	`redirect input`	Redireciona entrada de arquivo
`|`	`pipe`	Acopla saída e entrada

As opções de cada comando podem ser obtidas com `--help`

Compactação

- As ferramentas de compressão mais usadas no Linux são:
  - Tar: combina ou separa um arquivo em vários
  - Gzip: comprime e descomprime arquivos
  - São frequentemente combinadas

```
$ tar cvfz Lab01.tar.gz Lab01
```

Comprime a pasta Lab01 no arquivo .tar.gz

```
$ tar xvfz Lab01.tar.gz
```

Descomprime o arquivo lab01.tar.gz

Atalho	Descrição
x	e**x**tract (descomprimir)
c	**c**ompress (comprimir)
v	**v**erbose (exibe mensagens)
f	**f**orce (não pergunte)
z	Use o g**z**ip

Instalação de Aplicativos

- Arquivos .deb podem ser baixados e instalados
  - Diretamente pelo terminal (ou através de uma ferramenta gráfica):

```
$ sudo apt install
./file.deb
```

- A instalação pode ser feita também através dos repositórios
  - As distribuições mantêm estes repositórios online
  - A versão do aplicativo varia com cada distribuição

```
$ sudo apt install vim
```

Ferramentas

- Para criar compiladores vamos precisar:
  - Compilador e depurador C++ (g++ e gdb)
  - Sistema de automação da compilação (make e cmake)
  - Geradores de analisadores léxicos e sintáticos (flex e bison)

Instalando Aplicativos pelo Terminal

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install g++ gdb
$ sudo apt install make cmake
$ sudo apt install flex libfl-dev
$ sudo apt install bison libbison-
dev
```

Compilador C++

- GNU g++ é o compilador padrão do Linux

```
$ g++  parser.cpp postfix.cpp   –std=c++17   –o
postfix
```

  - Argumentos da linha de comando:
    - Nomes dos arquivos fonte (.cpp)
    - Padrão da linguagem a ser utilizado (C++17)
    - Nome do arquivo executável
      - Se não fornecido, gera um arquivo chamado a.out
  - Para depurar o código compile com a opção `-g` (Debug)
  - Para otimizar o código compile com a opção –02 (Release)

Depurador C++

- GNU gdb é o depurador padrão do sistema

```
$ g++ -g parser.cpp postfix.cpp –std=c++17 –o postfix

$ gdb postfix -tui
```

Comando	Atalho	Descrição
break	b	adiciona um ponto de parada (breakpoint)
run	r	executa o código até o primeiro ponto de parada
step	s	realiza um passo, entrando em funções (step into)
next	n	realiza um passo, saltando funções (step over)
print	p	exibe o valor de uma variável
quit	q	sai do depurador

A tecla ENTER repete o último comando.

Make

- O GNU Make é um sistema de automação da compilação
  - Simplifica o ciclo editar-compilar-executar
    - Compila com apenas um comando, make
    - É preciso criar um arquivo *makefile*

makefile

```
# variáveis
CPP=g++
ARGS=-g -std=c++17
# regras
all:
    $(CPP) postfix.cpp parser.cpp $(ARGS) -o
postfix
```

Arquivo *makefile* normalmente fica na pasta dos arquivo fonte (.cpp)

Make

- Um *makefile* mais complexo permite:
  - Especificação de dependências
  - Recompilação seletiva

Ao receber arquivos objeto, o compilador gera o executável apenas ligando os arquivos pré-compilados.

makefile

```
CPP=g++
ARGS=-c -g -std=c++17

all: postfix

postfix: postfix.o parser.o
    $(CPP) postfix.o parser.o -o
postfix

postfix.o: postfix.cpp parser.h
    $(CPP) $(ARGS) postfix.cpp

parser.o: parser.cpp parser.h
    $(CPP) $(ARGS) parser.cpp

clean:
    rm postfix postfix.o parser.o
```

CMake

- CMake é um gerador de sistemas de compilação
  - Os seguintes sistemas são suportados no Linux:
    - Unix Makefiles
    - Ninja
    - Watcom WMake
    - CodeBlocks - Ninja
    - CodeBlocks - Unix Makefiles
    - CodeLite - Ninja
    - CodeLite - Unix Makefiles
    - Sublime Text 2 - Ninja
    - Sublime Text 2 - Unix Makefiles
    - Kate - Ninja
    - Kate - Unix Makefiles
    - Eclipse CDT4 - Ninja
    - Eclipse CDT4 - Unix Makefiles
    - KDevelop3
    - KDevelop3 - Unix Makefiles

CMake

- As configurações são feitas em CMakeLists.txt
  - CMake gera um Makefile e vários arquivos auxiliares
  - Melhor executar a partir de uma pasta separada (Debug)

```
~/Lab01/Draft/Debug$ cmake ../
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.0.0)
project(Postfix)
set(CMAKE_BUILD_TYPE Debug)
set(CMAKE_CXX_STANDARD 17)
set(SOURCE_FILES parser.cpp postfix.cpp)
add_executable(postfix ${SOURCE_FILES})
```

```
+-- Draft/
|   +-- Debug/
|   +-- parser.cpp
|   +-- parser.h
|   +-- postfix.cpp
|   +-- CMakeLists.txt
```

CMake

- Para gerar configurações Debug e Release
  - Retire a opção CMAKE_BUILD_TYPE do arquivo

```
~/Lab01/Final/Debug$ cmake -DCMAKE_BUILD_TYPE=Debug ../

~/Lab01/Final/Release$ cmake -DCMAKE_BUILD_TYPE=Release ../
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.0.0)
project(Postfix)
#set(CMAKE_BUILD_TYPE Debug)
set(CMAKE_CXX_STANDARD 17)
set(SOURCE_FILES parser.cpp postfix.cpp)
add_executable(postfix ${SOURCE_FILES})
```

```
+-- Final/
|   +-- Debug/
|   +-- Release/
|   +-- parser.cpp
|   +-- parser.h
|   +-- postfix.cpp
|   +-- CMakeLists.txt
```

Resumo

- Vamos trabalhar no Linux
  - Usando o terminal
    - ls, mkdir, cd, rm, cat, <, >, |, etc.
  - Com a linguagem C++
    - Compilador: g++
    - Depurador: gdb
    - Automação da compilação: make
    - Gerador de Makefile: cmake