

Sistemas Operacionais II

Linux: Introdução



Histórico do UNIX

- **1969:** origem como um projeto de pesquisa do AT&T Bell Labs
- **1976:** Unix disponível gratuitamente nas universidades
- **1977:** Universidade de Berkeley licencia o código da AT&T e faz suas próprias versões do Unix: BSD

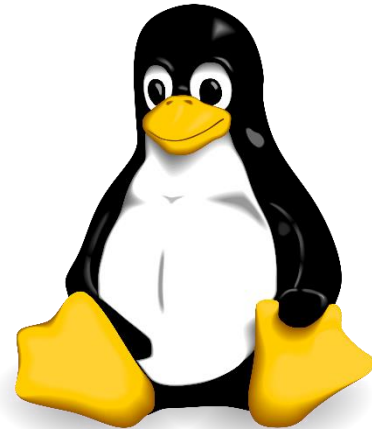


Histórico do UNIX

- Com aceitação comercial, preço das licenças da AT&T sobem rapidamente
- Berkeley decide eliminar código da AT&T de seu Unix, mas perde as verbas para pesquisa de SO antes de concluir o projeto, deixando apenas o BSD-Lite.
- BSD-Lite dá origem aos vários UNIX BSD: BSD/OS, FreeBSD, NetBSD e OpenBSD
- Demais UNIX (HP-UX, Solaris, IRIX) são descendentes da linhagem AT&T original.

Histórico do Linux

- Criado em 1991 por Linus Torvalds, aluno da Universidade de Helsinki, Finlândia.
 - O nome Linux provém de “Linus” e “UNIX”.
- O Minix serviu como inspiração para o projeto
- Torvalds buscou conselhos na comunidade.
- Os desenvolvedores continuaram a apoiar o conceito de um novo sistema operacional gratuito.

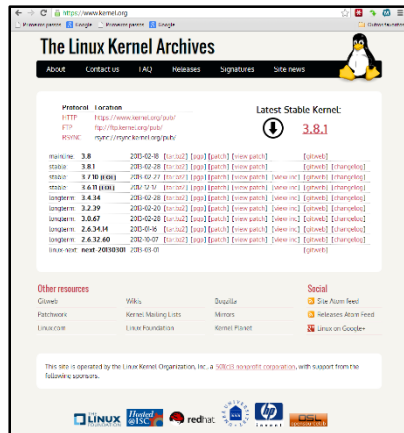


Histórico do Linux

- Kernel 1.0 lançado em 1994
- Versão atual (em 02/03/2020): 5.5.7



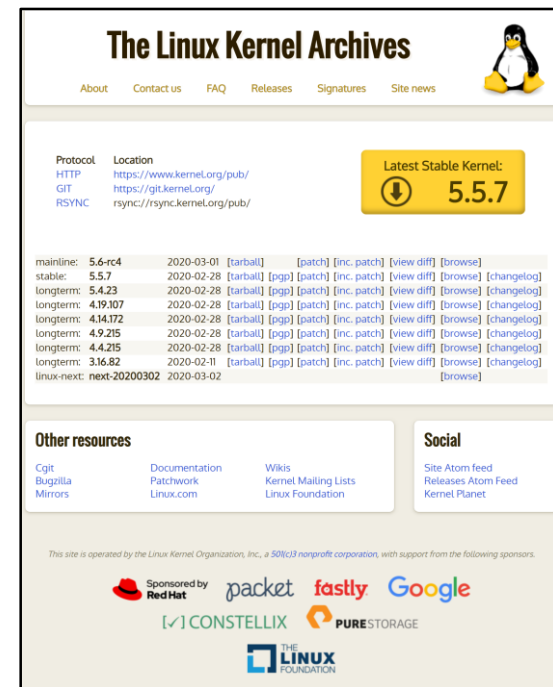
1S/2012 – 3.2.8



1S/2013 – 3.8.1



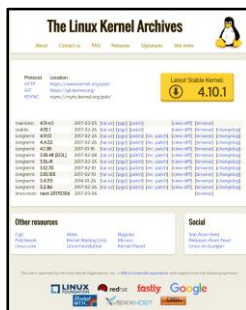
1S/2014 – 3.13.4



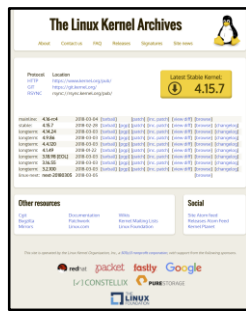
1S/2015 – 4.0



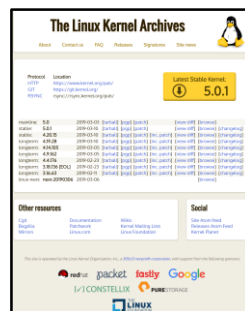
1S/2016 – 4.4.3



1S/2017 – 4.10.1



1S/2018 – 4.15.7



1S/2019 – 5.0.1

03/03/2020

Distribuição Linux

- Permite que os usuários não familiarizados com os detalhes do Linux instalem e usem o Linux.
- Inclui softwares como:
 - Núcleo do Linux.
 - Aplicações de sistema (por exemplo, gerenciamento de conta de usuário, gerenciamento de rede e ferramentas de segurança).
 - Aplicações de usuário (por exemplo, GUIs, navegadores Web, editores de texto, aplicações de e-mail, bancos de dados e jogos).
 - Ferramentas para simplificar o processo de instalação.

Visão geral do Linux

- Os sistemas Linux incluem, além do núcleo, interfaces com usuário e aplicações.
- Empréstimo do UNIX a abordagem em camadas do sistema.
- O sistema contém threads de núcleo para executar serviços.
 - Implementados como *daemons*, que permanecem adormecidos até serem acordados por um componente do núcleo.
- Sistema multiusuário
 - Restringe o acesso a operações importantes a um usuário com privilégios de superusuário (também chamado de raiz - root).

Desenvolvimento e comunidade

- Torvalds controla todas as modificações no núcleo.
- Ele conta com um grupo de mais ou menos 20 “tenentes” para gerenciar melhorias no núcleo.
- Quando o desenvolvimento do núcleo está para ser concluído:
 - Congelamento de característica: nenhuma característica nova é adicionada ao núcleo.
 - Congelamento de código: apenas os códigos que corrigem erros importantes são aceitos.
- Várias corporações apoiam o desenvolvimento do Linux.
- O Linux é distribuído sob a GNU Public License (GPL).
- O Linux é um software gratuito protegido por direitos autorais.

Distribuições Linux

Distribuição	Site Web
CentOS	www.centos.org
Debian	www.debian.org
Fedora	getfedora.org
Gentoo	www.gentoo.org
Mint	www.linuxmint.com
openSUSE	www.opensuse.org
Red Hat Enterprise	www.redhat.com
Slackware	www.slackware.com
SUSE Linux Enterprise	www.suse.com
TurboLinux	www.turbolinux.com
Ubuntu	www.ubuntu.com
Mageia	www.mageia.org



Red Hat

- **Red Hat Linux:** uma das primeiras distribuições Linux, primeira a implementar pacotes RPM, hoje descontinuada em favor da versão comercial.
- **Red Hat Enterprise Linux:** versão comercial, com assinatura anual (suporte pago)
- **Fedora:** substituiu a versão doméstica do Red Hat, projeto patrocinado pela Red Hat

CentOS



- Derivado do código-fonte do Red Hat Enterprise Linux
 - Empresa é obrigada a liberar os fontes por conta das licenças (principalmente a GNU)
- Não contém as ferramentas proprietárias da Red Hat
- Não contém os logotipos da Red Hat
- Boa escolha para quem quer uma distribuição orientada a produção sem pagar para a Red Hat
- Desde 2014, a Red Hat passou a financiar o projeto.

SUSE Linux



- Bastante forte na Europa
- Produzido na Alemanha
- Comprado pela Novell em 2004
- Tem hoje versão grátis disponível na web e versão comercial mais completa
 - SUSE Linux Enterprise
 - openSUSE

Debian

- Distribuição mais ligada ao projeto GNU
- Não é uma empresa, mas sim um projeto de mais de 1000 desenvolvedores voluntários em todo o mundo
- É base de outras distribuições: Knoppix e Ubuntu



Ubuntu

- Derivado do Debian
- Focado em usabilidade
- Gratuito
- Patrocinado pela empresa Canonical do Reino Unido, cujo dono é o empresário sul-africano Mark Shuttleworth
 - Canonical obtém lucros vendendo suporte técnico



Mint



- Baseado no Ubuntu
 - Inicialmente tinha o objetivo de ser mais completo logo que instalado, incluindo plugins de navegadores, codecs de mídia, suporte para reprodução de DVDs, Java, etc.
 - A partir da versão 18 essas ferramentas podem ser instaladas pelo usuário após o sistema estar em execução.

Slackware

- A mais antiga distribuição Linux que ainda é mantida
- Já foi dominante, mas hoje é menos popular
- Simples, rápido e estável, ao custo de ser menos amigável (mais difícil de aprender)

The logo for Slackware Linux. It features the word "slackware" in a large, bold, lowercase serif font. Below it, the word "linux" is written in a smaller, lowercase monospace font. A horizontal line is positioned under "slackware", starting from the left edge of the letter 's' and extending to the right edge of the letter 'e'. The word "linux" is positioned to the right of this line, with its first letter 'l' aligned with the end of the line.

Interface com o usuário

- Pode ser acessada por interpretadores de comando de console, como bash, csh e esh.
- A maioria das GUIs do Linux é composta de diversas camadas:
 - X Window System
 - Nível mais inferior.
 - Oferece mecanismos de camadas GUI mais altas para criar e manipular componentes gráficos.
 - Gerenciador de janela
 - Aproveita mecanismos da interface X Window System para controlar a busca, a aparência, o tamanho e outros atributos de janelas.
 - Ambiente de mesa (por exemplo, KDE, GNOME)
 - Oferece aplicações e serviços ao usuário.

Padrões

- Cada vez mais o Linux atende a padrões populares como o POSIX.
 - POSIX (Portable Operating System Interface): família de padrões definida pelo IEEE para manter compatibilidade entre sistemas operacionais
 - POSIX define uma interface de programação de aplicações (API), junto com shells de linha de comando e interfaces de utilitários, para compatibilidade de software entre variantes do Unix e outros sistemas operacionais.

Padrões

- Single UNIX Specification (SUS)
 - Conjunto de padrões que definem interfaces de programação de usuário e aplicação para sistemas operacionais UNIX, interpretadores de comandos e utilidades.
 - A versão 3 do SUS associa vários padrões (incluindo POSIX, os padrões ISO e versões anteriores do SUS).
- Linux Standards Base (LSB)
 - Projeto que busca padronizar o Linux de modo que as aplicações escritas para uma distribuição que siga o LSB compilem e comportem-se exatamente como em qualquer outra distribuição que também siga o LSB.

Buscando informações: man e info

- Páginas de manual
 - man *nome_do_comando*
- Documentos Texinfo
 - info *nome_do_comando*
 - criado pois o comando *nroff* que formata páginas de manual é proprietário da AT&T (hoje já existe a versão GNU *groff*)
 - definem um segundo padrão desnecessário e complicado

Páginas de Manual

- Distribuições do Linux incluem páginas de manual para maioria dos comandos, chamadas de sistema, e funções da biblioteca padrão
- As páginas são divididas em seções; para programadores, as mais importantes são:
 1. Comandos de Usuário
 2. Chamadas de Sistema
 3. Funções da Biblioteca Padrão
 4. Comandos de Sistema/Administrativos
- Exemplos:
 - **man sleep** → manual do comando sleep
 - **man 3 sleep** → manual da função sleep da biblioteca padrão

Mais informações

- Arquivos de cabeçalho em **/usr/include**
 - Útil para verificar erros em chamadas de sistema
 - Verificar se assinatura da função corresponde ao manual
 - Eventualmente a página de manual pode estar desatualizada
- Código-Fonte
 - O Linux tem código aberto e disponível livremente, podendo ser consultado.
 - O código-fonte pode não estar instalado no sistema por padrão.
 - O código fonte do núcleo normalmente está em **/usr/src/linux**

Localizar e instalar software: *which*

- Para saber se um aplicativo está no seu sistema use:
 - *which nome_do_aplicativo*
 - Exemplo:
 which httpd
 /usr/sbin/httpd
 - Busca no caminho de pesquisa (PATH)

Localizar e instalar software:

whereis e locate

- Whereis httpd
 - Intervalo de diretórios mais amplo
- Locate httpd
 - Pesquisa em um índice pré-compilado do sistema de arquivos
 - O banco de dados é construído através do comando *updatedb*
 - Normalmente executado diariamente pelo *cron*

Localizar e instalar software (Red Hat, Fedora, SuSE, etc.)

- `rpm -q python`
 - Verifica se o pacote chamado python está instalado
 - Pacotes RPM são utilizados no Red Hat, Fedora, SuSe, etc...
 - Normalmente é mais fácil instalar o pacote binário compilado específico para sua distribuição
 - Você também pode buscar e compilar o código-fonte original
- `dnf install python`
 - Obtém e instala a versão mais recente do Python
 - DNF é um gerenciador de pacotes que usa o Sistema RPM
 - É um fork do YUM, resolvendo alguns problemas do mesmo



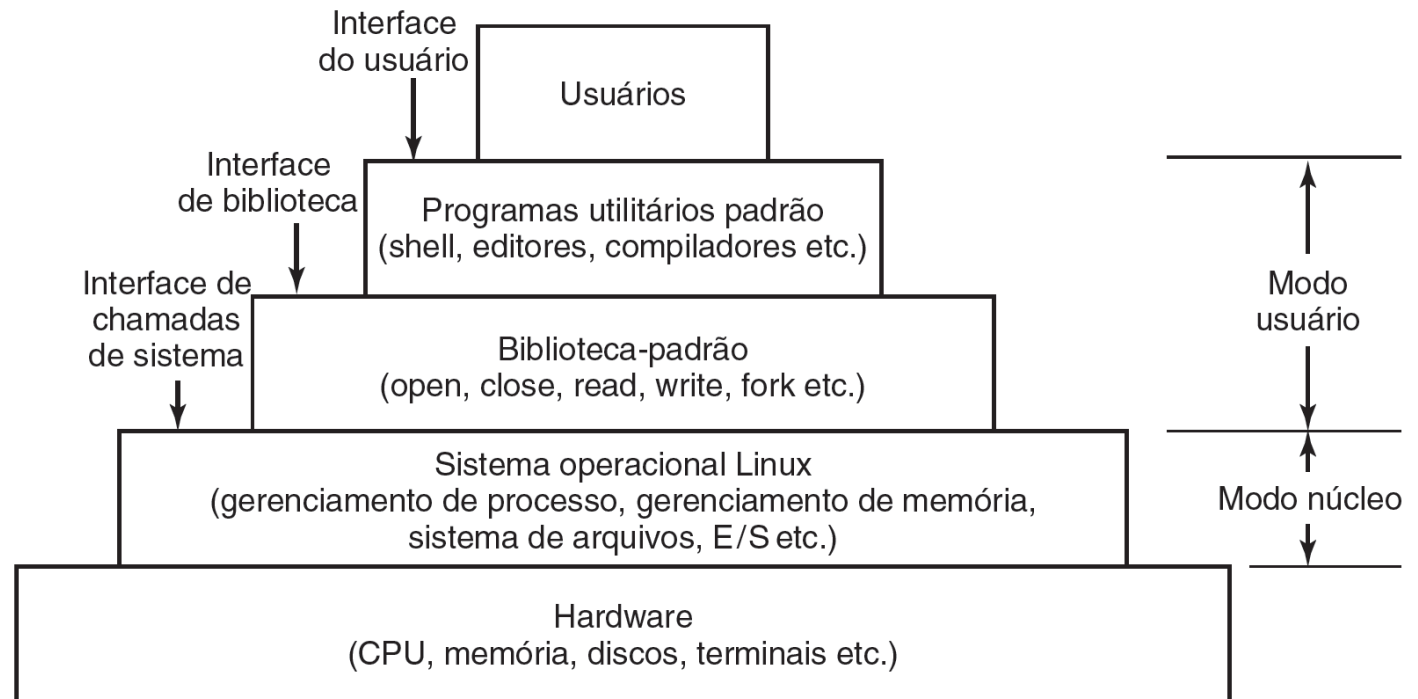
Localizar e instalar software: (Debian, Ubuntu, etc.)

- `apt-get install python`
 - Obtém e instala a versão mais recente do Python
 - Utilizado no Debian e seus derivados, como o Ubuntu e outros



ubuntu

Interfaces para o Linux



■ **Figura 10.1** As camadas em um sistema Linux.

Programas utilitários do Linux

- Categorias dos programas utilitários:
 - Comandos de manipulação de arquivos e diretórios.
 - Filtros.
 - Ferramentas de desenvolvimento de programas, como editores e compiladores.
 - Processamento de texto.
 - Administração de sistema.
 - Miscelâneas.

Programa	Tipicamente
cat	Concatena vários arquivos para a saída-padrão
chmod	Altera o modo de proteção do arquivo
cp	Copia um ou mais arquivos
cut	Corta colunas de texto de um arquivo
grep	Procura um certo padrão dentro de um arquivo
head	Extraí as primeiras linhas de um arquivo
ls	Lista diretório
make	Compila arquivos e constrói um binário
mkdir	Cria um diretório
od	Gera uma imagem (dump) octal de um arquivo
paste	Cola colunas de texto dentro de um arquivo
pr	Formata um arquivo para impressão
ps	Lista os processos em execução
rm	Remove um ou mais arquivos
rmdir	Remove um diretório
sort	Ordena um arquivo de linhas alfabeticamente
tail	Extraí as últimas linhas de um arquivo
tr	Traduz entre conjuntos de caracteres

Tabela 10.1 Alguns dos programas utilitários Linux comuns, requeridos por POSIX.

Interpretador de Comandos (Shell)

- Bash é o mais comum e padrão na maioria dos sistemas
 - BASH é um acrônimo de Bourne Again Shell
 - Bourne é a shell original do UNIX
- Shell é um programa comum de usuário
 - Quando usuário digita uma linha de comando:
 - Extrai a primeira palavra, procura pelo programa correspondente e o executa
 - Suspende a si próprio até que o programa executado termine

Interpretador de Comandos (Shell)

- Comandos podem levar argumentos que são passados como cadeias de caracteres ao programa chamado.
 - Exemplos:
 - `cp src dest` (dois argumentos)
 - `head -20 file` (flag indicada com um traço)
 - `ls *.c` (wildcards)

Interpretador de Comandos (Shell)

- `sort` (Ctrl+D para fim de arquivo)
- `sort <in >out` (lê entrada de in e escreve em out)
- `sort <in >temp; head -30 <temp; rm temp`
(lê entrada de in, grava saída em temp, imprime primeiras 30 linhas de temp e remove temp)
- `sort <in | head -30`
(igual exemplo anterior, mas cria uma pipeline, dispensando arquivo temporário)
- `grep ter *.t | sort | head -20 | tail -5 > foo`
[pega todas as linhas com 'ter' de arquivos .t, ordena-as, pega as 20 primeiras linhas da lista ordenada, pega as 5 últimas linhas (linhas 16-20 da lista anterior) e grava em *foo*]

Interpretador de Comandos (Shell)

- `wc -l <a >b &`

(wc – contador de palavras, flag -l indica que deve contar linhas, *a* é entrada, *b* é saída, tudo feito em segundo plano por causa do &)

- `sort <x | head -1 &`

(pipeline em segundo plano)

Interpretador de Comandos (Shell)

- É possível criar um arquivo texto com vários comandos de *shell* (*shell script*) e depois executá-lo chamando o *shell* com tal arquivo como entrada
 - Também é possível usar estruturas *if*, *for*, *while*, *case*, etc.
 - Berkeley C é um *shell* alternativo com sintaxe semelhante à linguagem C
 - Qualquer pessoa pode escrever seu próprio *shell* dando suporte a outras linguagens

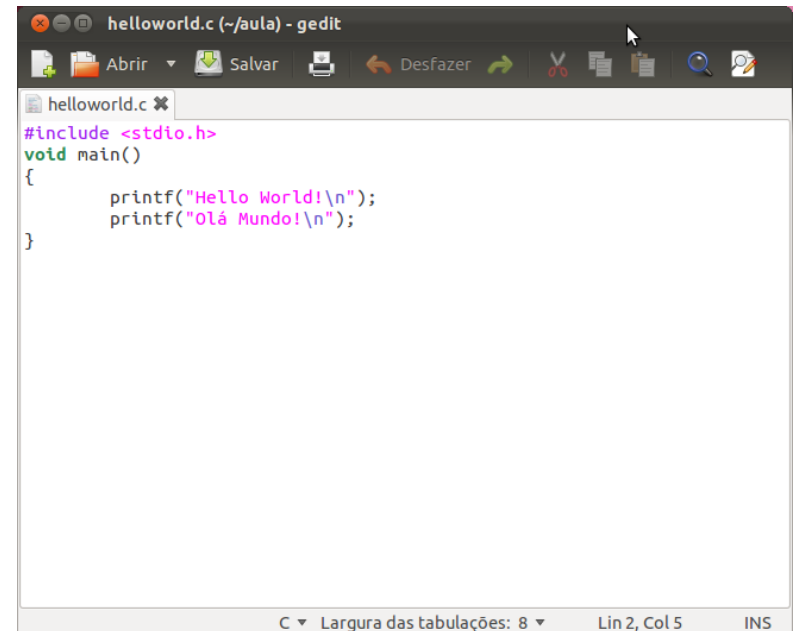
Editores de Texto

- **Modo Texto**
 - **vi**
 - Amplamente disponível
 - Difícil aprendizado
 - **vim**
 - Versão melhorada do vi
 - **pico**
 - Aprendizado mais fácil
 - Parte do **pine**
 - Destaque de sintaxe de C e outras linguagens
 - **nano**
 - Clone do pico, sem a sobrecarga do pine

[illegible]

Editores de Texto

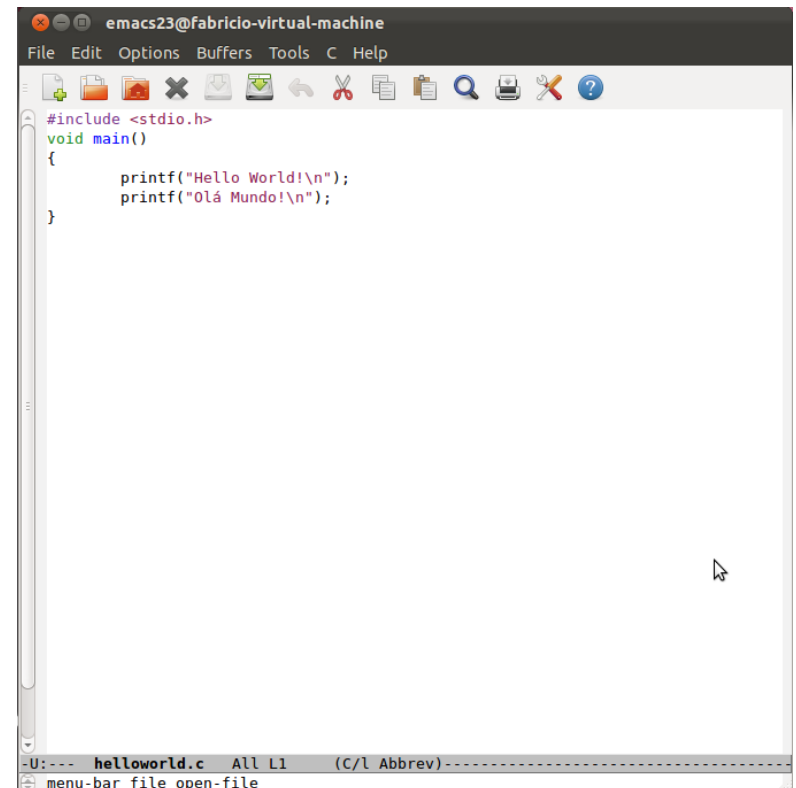
- Modo Gráfico
 - **gedit**
 - Editor padrão da interface Gnome
 - **Kate**
 - Editor padrão da interface KDE
 - **Emacs**
 - Bastante poderoso e personalizável



The screenshot shows the gedit text editor window titled "helloworld.c (~/.aula) - gedit". The menu bar includes "Abrir", "Salvar", "Desfazer", and others. The code in the editor is:

```
#include <stdio.h>
void main()
{
    printf("Hello World!\n");
    printf("Olá Mundo!\n");
}
```

The status bar at the bottom indicates "C", "Largura das tabulações: 8", "Lin 2, Col 5", and "INS".

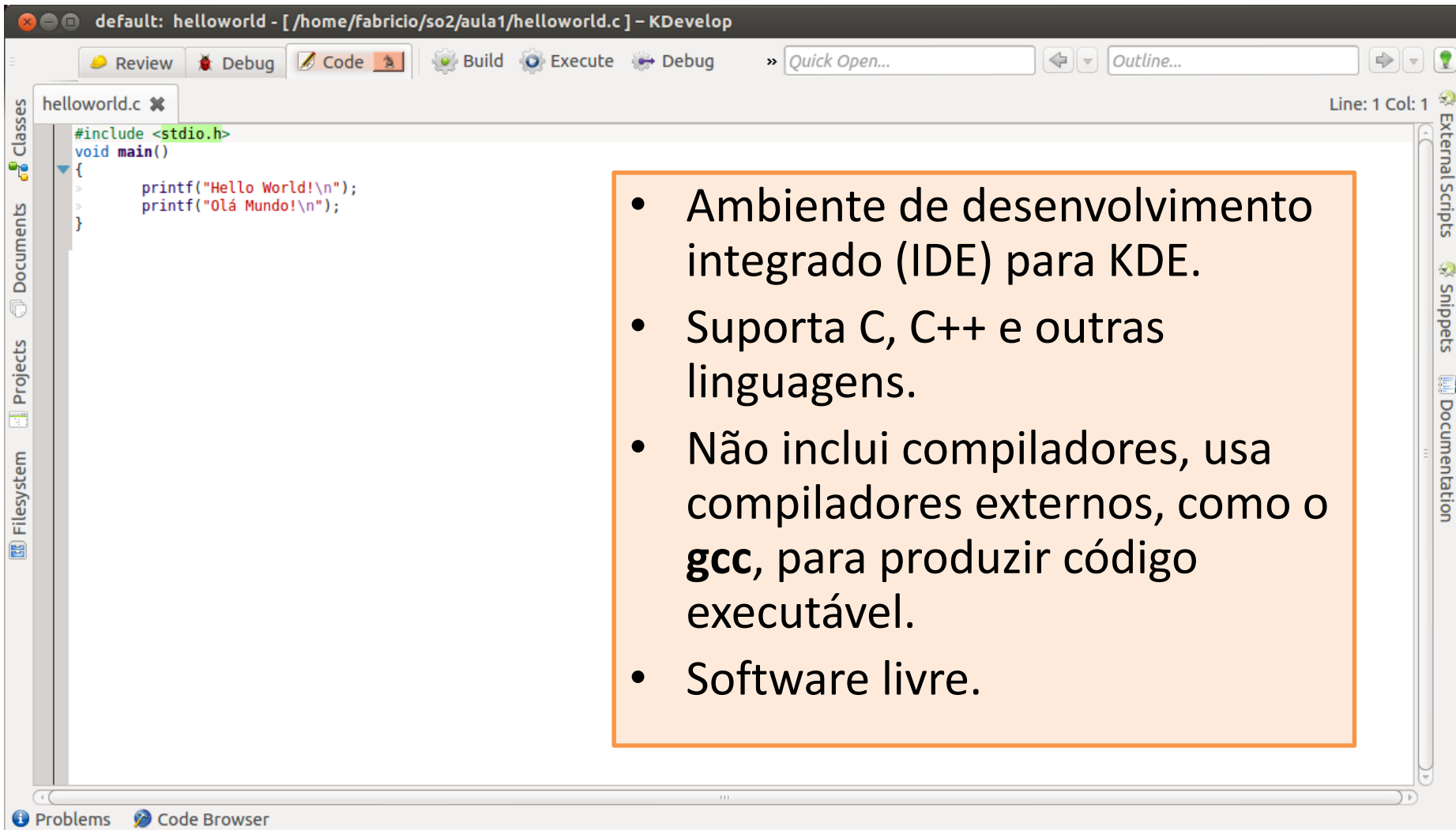


The screenshot shows the Emacs text editor window titled "emacs23@fabricio-virtual-machine". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "C", and "Help". The code in the editor is:

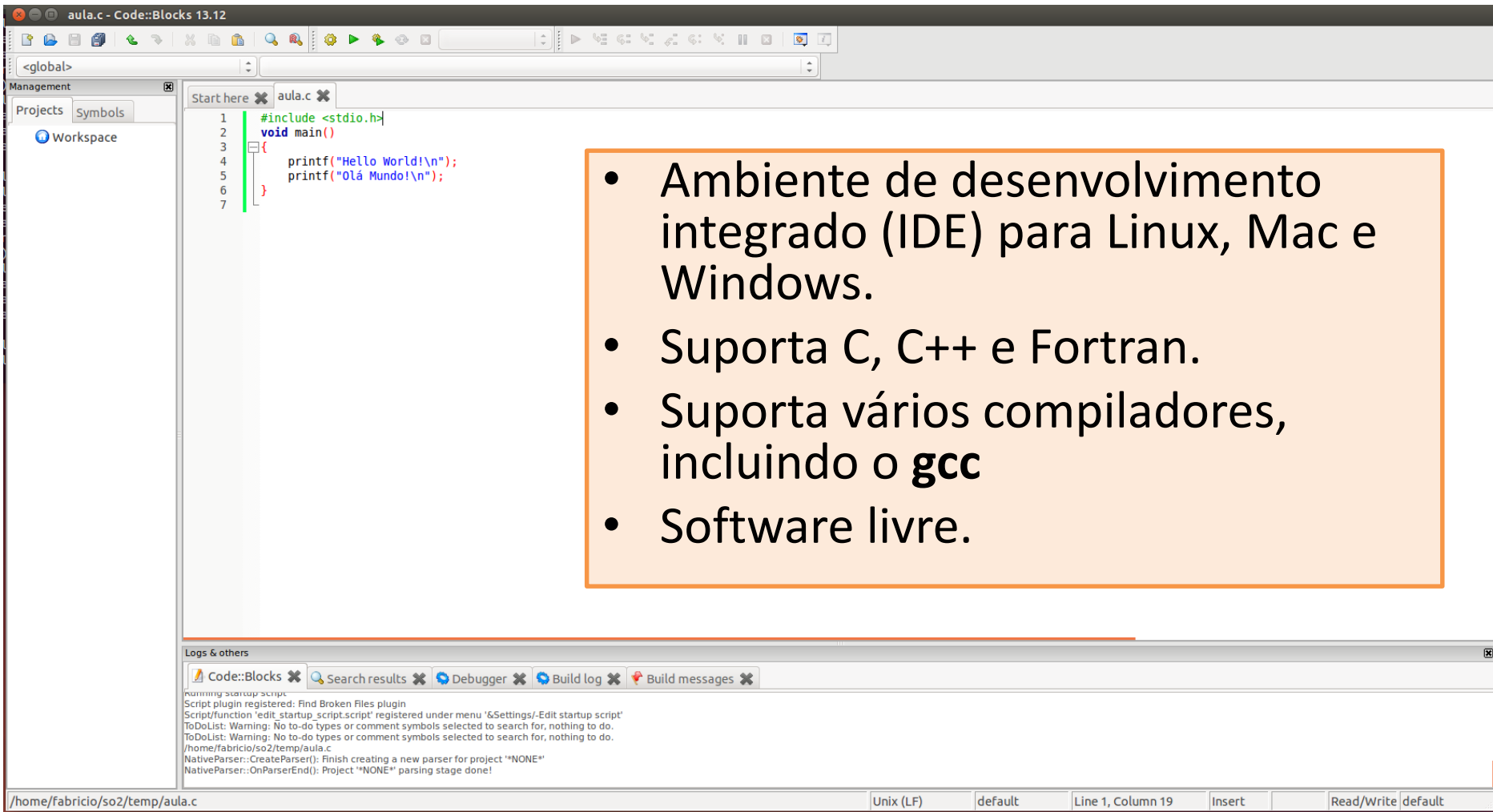
```
#include <stdio.h>
void main()
{
    printf("Hello World!\n");
    printf("Olá Mundo!\n");
}
```

The status bar at the bottom shows the file path "helloworld.c" and other details.

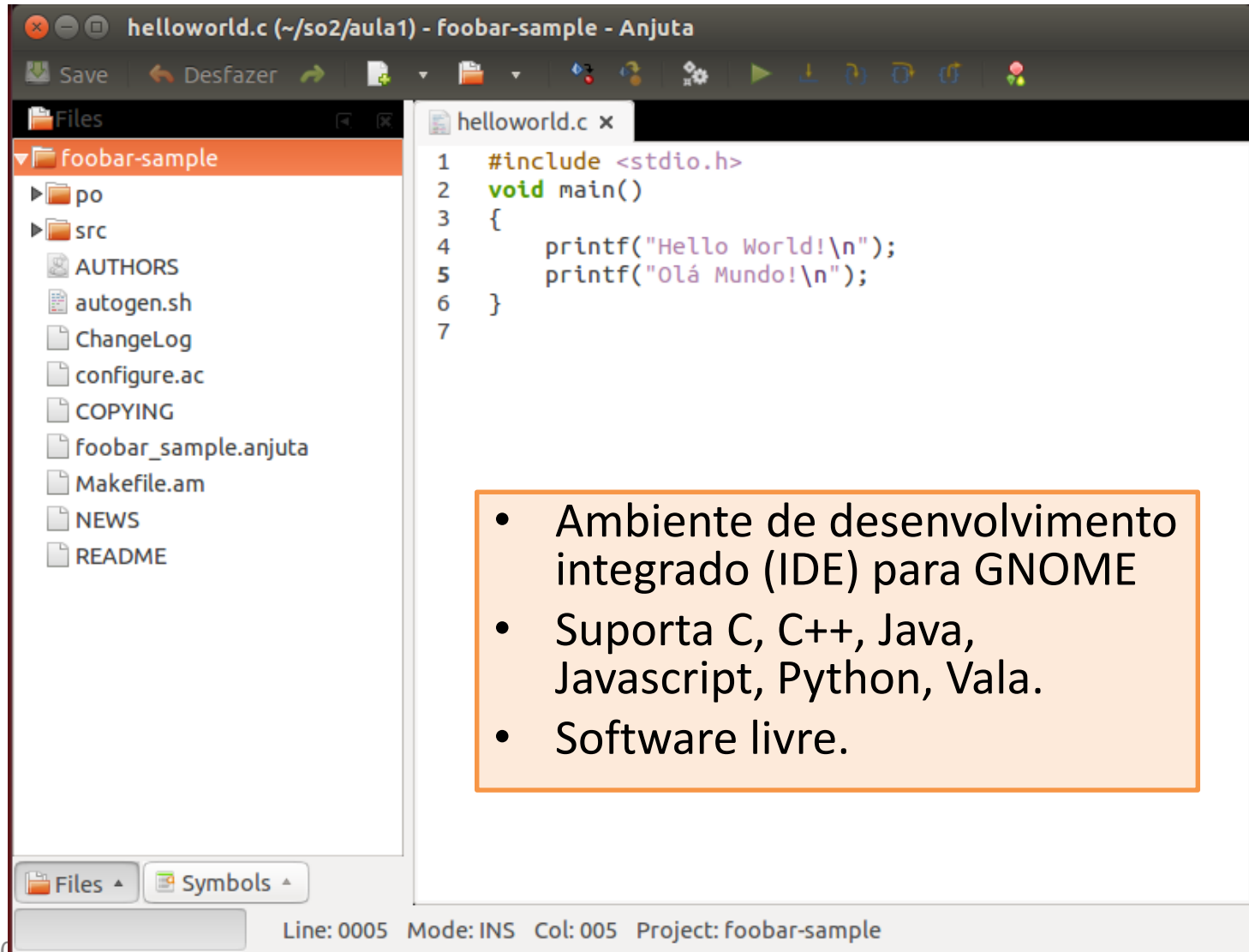
KDevelop



Code::Blocks



Anjuta DevStudio



Outros editores e ambientes de desenvolvimento

- Netbeans
- Eclipse
- CodeLite IDE
- Atom
- Sublime Text
- Bluefish Editor
- Brackets
- Geany IDE

Programando

- Use seu editor de textos preferido para criar código-fonte na linguagem:
 - C
 - *.c* ou *.h*
 - C++
 - *.cpp*, *.hpp*, *.cxx*, *.hxx*, *.C*, *.H*
 - *Etc...*

Compiladores

- GNU Compiler Collection (GCC)
 - C, C++, Java, Objective-C, Fortran, Ada, Go
 - gcc helloworld.c
 - Compila e linka, gerando executável
 - gcc -c main.c
 - Apenas compila código C, gerando arquivo objeto
 - g++ -c reciprocal.cpp
 - Apenas compila código C++, gerando arquivo objeto
 - g++ -c -I ../include reciprocal.cpp
 - Instrui o compilador a procurar cabeçalhos também em ../include

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "reciprocal.hpp"
int main (int argc, char **argv)
{
    int i;
    i = atoi (argv[1]);
    printf ("O inverso de %d é %g\n", i, reciprocal (i));
    return 0;
}
```

reciprocal.cpp

```
#include <cassert>
#include "reciprocal.hpp"
double reciprocal (int i)
{
    // deve ser diferente de zero
    assert (i != 0);
    return 1.0/i;
}
```

reciprocal.hpp

```
#ifdef __cplusplus
extern "C" {
#endif

extern double reciprocal(int i);

#ifdef __cplusplus
}
#endif
```

Compiladores

- `g++ -c -O2 reciprocal.cpp`
 - Compila com nível de otimização 2 (para código de produção)
- `g++ -o reciprocal main.o reciprocal.o`
 - Linka os objetos .o gerando o arquivo executável reciprocal
- `./reciprocal 7`
 - Executa o programa compilado, passando 7 como parâmetro

Compiladores

- `g++ -o reciprocal main.o reciprocal.o -lpam`
 - Inclui a biblioteca Pluggable Authentication Module (PAM) na linkagem
 - O compilador coloca o prefixo `lib` e o sufixo `.a` automaticamente
 - A busca por bibliotecas é feita nos locais padrão, como `/lib` e `/usr/lib`
- `g++ -o reciprocal main.o reciprocal.o -L /usr/local/lib/pam -lpam`
 - Especifica um caminho adicional para procurar por bibliotecas
- `g++ -o app app.o -L. -ltest`
 - É preciso indicar que a busca também deve ser feita no diretório atual, caso haja bibliotecas nele

Compiladores

- Você pode criar um arquivo Makefile para indicar dependências e automatizar a compilação com **make**

```
reciprocal: main.o reciprocal.o
    g++ $(CFLAGS) -o reciprocal main.o reciprocal.o

main.o: main.c reciprocal.hpp
    gcc $(CFLAGS) -c main.c

reciprocal.o: reciprocal.cpp reciprocal.hpp
    g++ $(CFLAGS) -c reciprocal.cpp

clean:
    rm -f *.o reciprocal
```

Compiladores

- `make`
 - Checa dependências em Makefile, compila, linka e gera executável
- `make clean`
 - Remove os arquivos objetos
- `make CFLAGS=-O2`
 - Passa a flag para o compilador, gerando executável otimizado

GNU Debugger (GDB)

- `make CFLAGS=-g`
 - `gcc -g -c main.c`
 - `g++ -g -c reciprocal.cpp`
 - `g++ -g -o reciprocal main.o reciprocal.o`
 - Gera executável com informações de depuração
- `gdb reciprocal`
 - Entra no depurador
 - Siga roteiro do capítulo 1.4.2 de Mitchell et al. [3] para testar o depurador

Referências Bibliográficas

1. [NEMETH, Evi.; SNYDER, Garth; HEIN, Trent R.; *Manual Completo do Linux: Guia do Administrador*. São Paulo: Pearson Prentice Hall, 2007. Cap. 1.](#)
2. [DEITEL, H. M.; DEITEL, P. J.; CHOFFNES, D. R.; *Sistemas Operacionais: terceira edição*. São Paulo: Pearson Prentice Hall, 2005. Cap. 20.](#)
3. [MITCHELL, Mark; OLDHAM, Jeffrey; SAMUEL, Alex; *Advanced Linux Programming*. New Riders Publishing: 2001. Cap. 1.](#)
4. [TANENBAUM, Andrew S.; BOS, Herbert. *Sistemas Operacionais Modernos*. 4ed. São Paulo: Pearson Education do Brasil, 2016. Cap. 10.](#)

