

# **SISTEMAS OPERACIONAIS**

Prof. Marcio Ponciano



# **Introdução a Sistemas Operacionais**

# Contextualização

Qual a importância de estudar sistemas operacionais no mundo atual



- Todos os computadores utilizam SO: computadores em geral, desktops, notebooks, embarcados, mobile, todos utilizam algum sistema operacional. Alguns sistemas são abertos, outros são proprietários.
- Permite entender o funcionamento do computador: o sistema operacional é responsável pelo comportamento do computador e dispositivos que são usados, e estudá-lo permitirá compreender como isso ocorre.
- Auxilia compreensão da computação em nuvem: um sistema operacional pode funcionar em máquinas virtuais tanto dentro de datacenter como em nuvem. A sua compreensão é essencial para mensurar o seu uso no mercado de trabalho.

# O que é um Sistema Computacional? Ou, simplesmente, um computador ou um Computador Moderno (Tanenbaum):



## Computador Moderno (Tanenbaum):

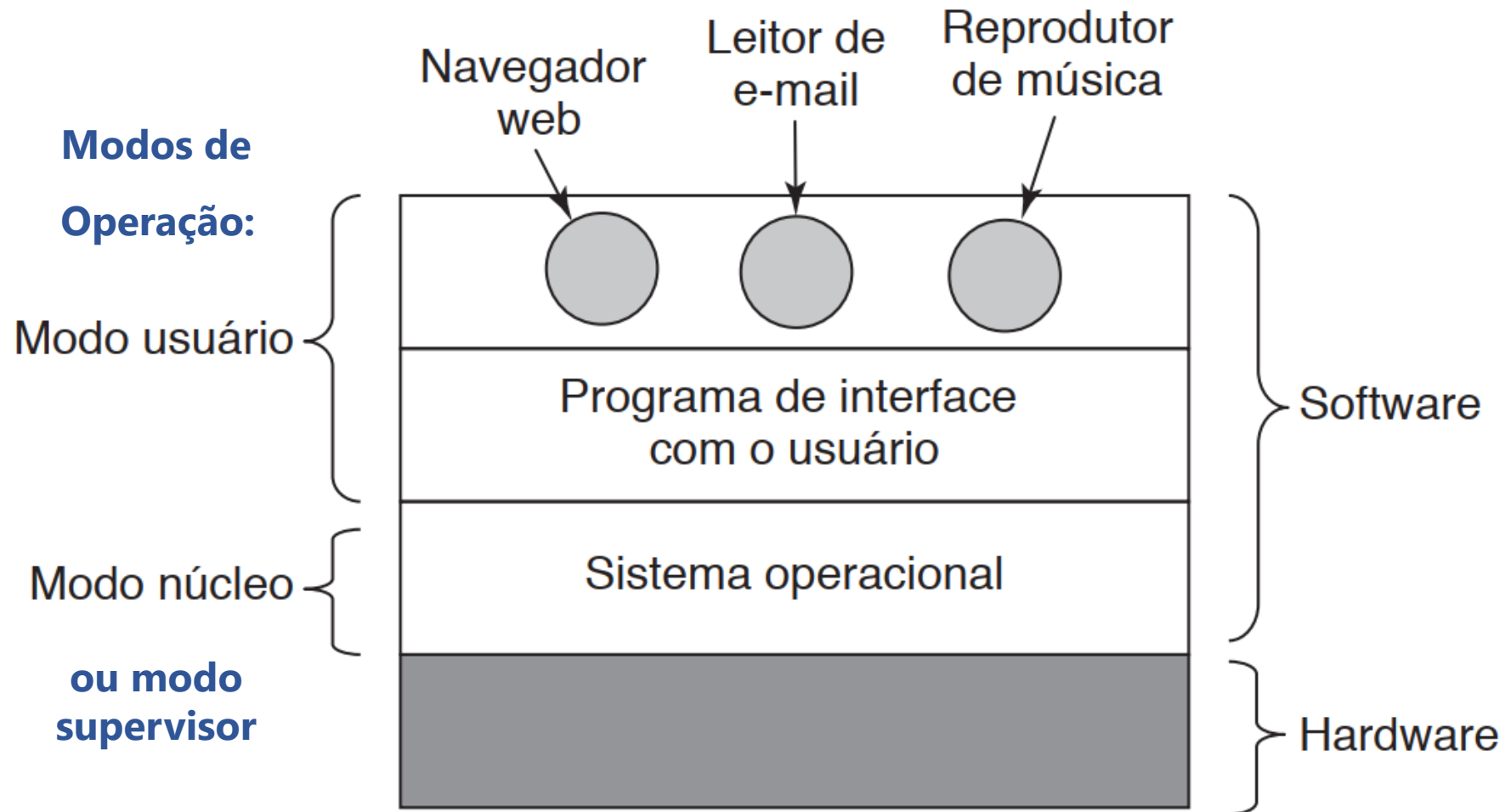
1. Um ou mais processadores
2. Memória principal
3. Discos, impressoras, teclado, mouse, monitor, interface de rede e outros dispositivos de entrada e saída (E/S)

“gerenciar todos esses componentes e usá-los de maneira otimizada é um trabalho extremamente desafiador. Por essa razão, computadores são equipados com um dispositivo de software chamado de **sistema operacional**” (Tanenbaum)

Software que gerencia todos os componentes de um computador e usá-los de maneira otimizada.



# Onde está o Sistema Operacional?



# Modos de execução

(**Modo Núcleo:** Nesse modo ele tem acesso completo a todo o hardware e pode executar qualquer instrução que a máquina for capaz de executar

**Modo usuário:** O resto do software opera em modo usuário, no qual apenas um subconjunto das instruções da máquina está disponível.

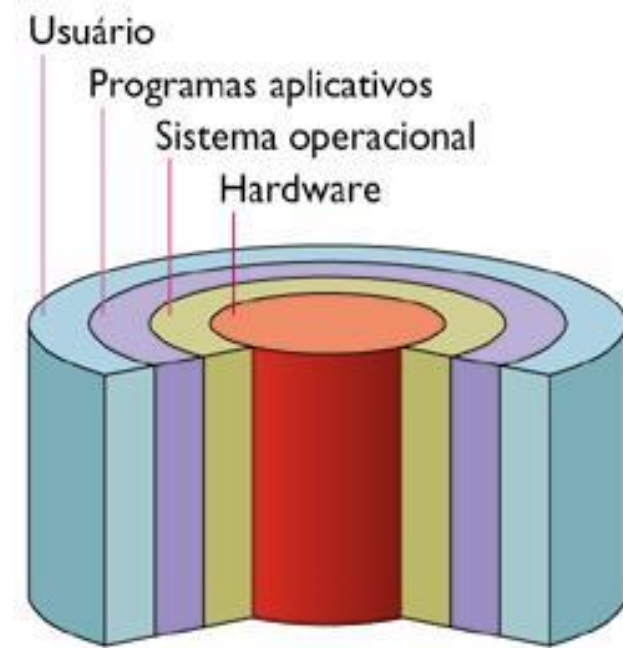
Em particular, aquelas instruções que afetam o controle da máquina ou realizam E/S (Entrada/Saída) são proibidas para programas de modo usuário





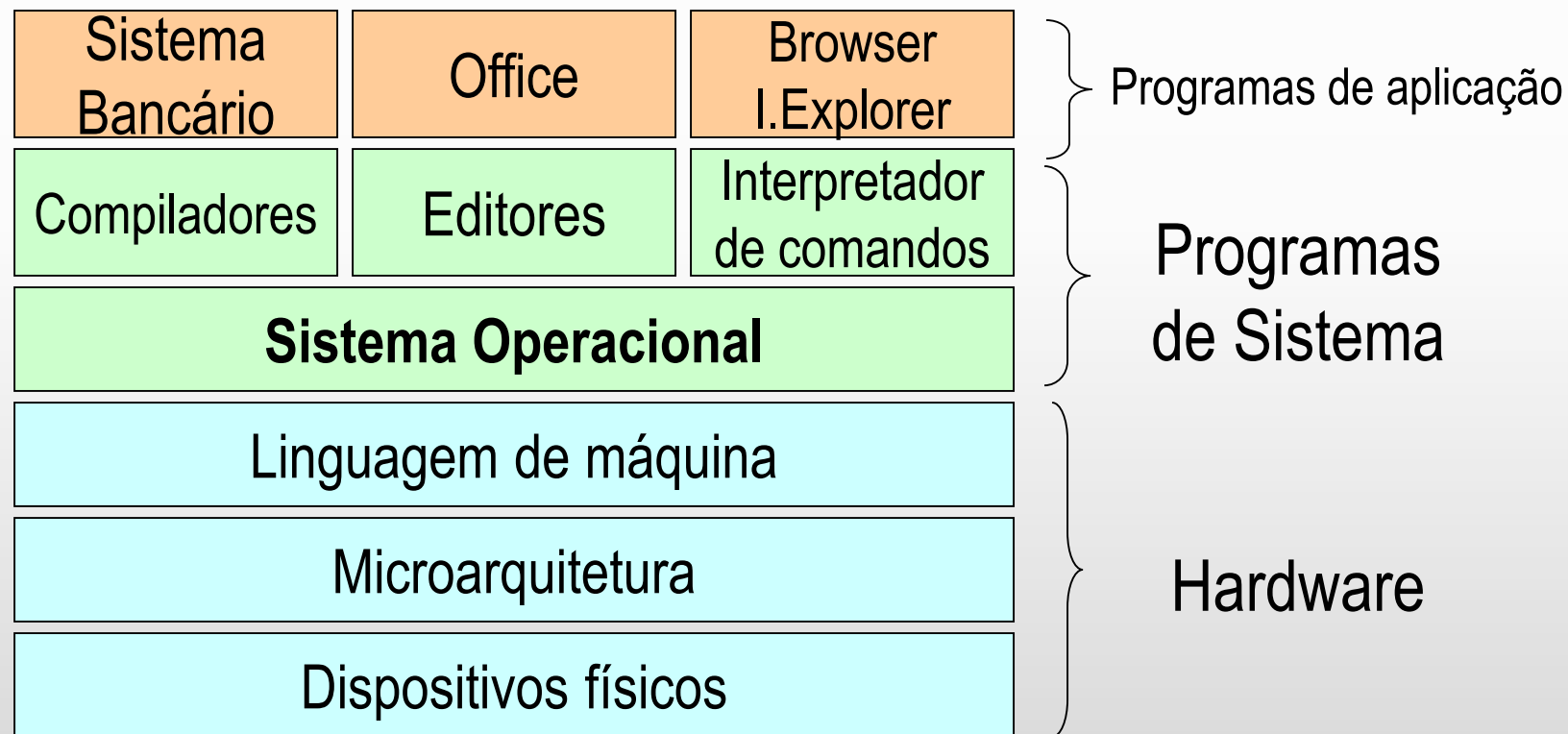
# Sistema complexo

- Um **conjunto** de programas que se situa entre os softwares aplicativos e o hardware:
  - Gerencia os recursos do computador (CPU, Memória, dispositivos periféricos).
  - Estabelece uma interface com o usuário.
  - Executa e oferece recursos para softwares aplicativos.



# Camadas

Camada de *software* entre o *hardware* e as aplicações dos usuários



```
self.file = None
self.fingerprints = self()
self.logdups = True
self.debug = debug
self.logger = logging.getLogger(__name__)
if path:
    self.file = os.path.join(path,
    self.file.seek(0)
    self.fingerprints.update(self)

classmethod
def from_settings(cls, settings):
    debug = settings.getbool(
    return cls(job_dir(settings), debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

def request_fingerprint(self, request):
    return request_fingerprint(request)
```



# E a BIOS, o que é?

Sistema Básico de entrada e saída (Basic Input Output System)

## Mas onde ele fica?

Na placa principal (placa-mãe)

## Funções:

conta com rotinas de E/S

## Armazenamento de dados:

Memória flash RAM, não volátil, mas permite atualizações

**BIOS NÃO É SO**



# Exemplo de BIOS

PhoenixBIOS Setup Utility				
Main	Advanced	Security	Boot	Exit
System Time: [10:34:40]			Item Specific Help	
System Date: [02/04/2016]				
Legacy Diskette A: [1.44/1.25 MB 3½"]			<Tab>, <Shift-Tab>, or <Enter> selects field.	
Legacy Diskette B: [Disabled]				
▶ Primary Master [None]				
▶ Primary Slave [None]				
▶ Secondary Master [None]				
▶ Secondary Slave [None]				
▶ Keyboard Features				
System Memory: 640 KB				
Extended Memory: 1047552 KB				
Boot-time Diagnostic Screen: [Disabled]				

F1	Help	↑↓	Select Item	-/+	Change Values	F9	Setup Defaults
Esc	Exit	↔	Select Menu	Enter	Select ▶ Sub-Menu	F10	Save and Exit

```
self.file = None
self.fingerprints = self()
self.logdupes = True
self.debug = debug
self.logger = logging.getLogger(__name__)
if path:
    self.file = os.path.join(path,
    self.file.seek(0)
    self.fingerprints.update(self)

classmethod
def from_settings(cls, settings):
    debug = settings.getbool('debug')
    return cls(job_dir(settings), debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

def request_fingerprint(self, request):
    return request_fingerprint(request)
```

# Comunicação com o usuário

Modos de comunicação:

- GUI - Graphical User Interface
- TUI - Text User Interface
- CUI/CLI - Command Line Interface

```
self.file = None
self.fingerprints = self()
self.logdups = True
self.debug = debug
self.logger = logging.getLogger(__name__)
if path:
    self.file = os.path.join(path,
    self.file.seek(0)
    self.fingerprints.update(self)

classmethod
def from_settings(cls, settings):
    debug = settings.getbool
    return cls(job_dir(settings), debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

def request_fingerprint(self, request):
    return request_fingerprint(request)
```



# Comunicação com o usuário

## GUI - Graphical User Interface

- O usuário se comunica com o SO através de sua interface que pode ser:
  - Gráfica: GUI (Graphical User Interface)
    - Composto por menus, ícones, janelas...

```
self.file = None
self.fingerprints = self()
self.logdupes = True
self.debug = debug
self.logger = logging.getLogger(__name__)
if path:
    self.file = os.path.join(path,
    self.file.seek(0)
    self.fingerprints.update({key: value})

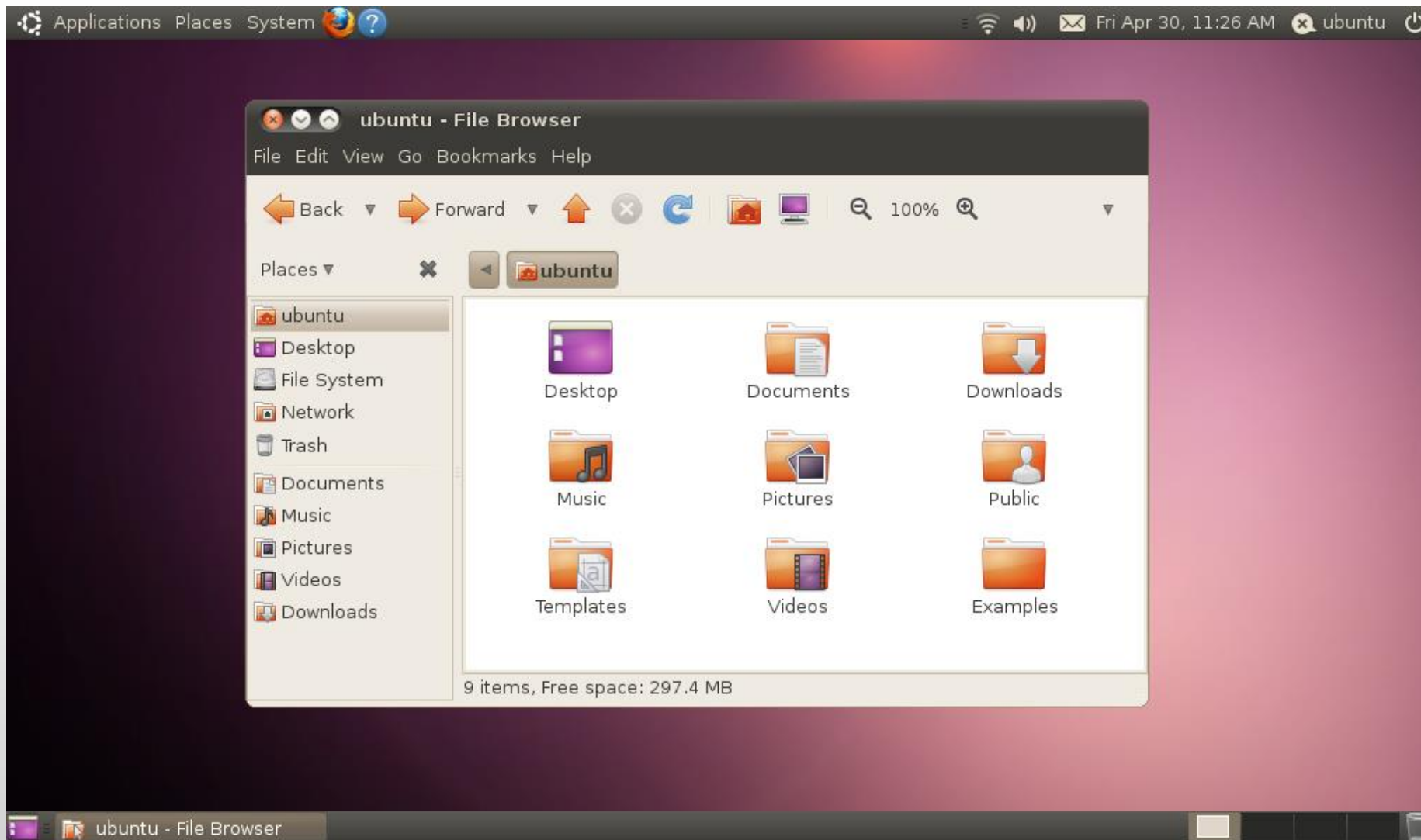
classmethod
def from_settings(cls, settings):
    debug = settings.getbool('debug')
    return cls(job_dir(settings), debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

def request_fingerprint(self, request):
    return request_fingerprint(request)
```

# Comunicação com o usuário

## Exemplo de interface GUI



```
self.file = None
self.fingerprints = self()
self.logdups = True
self.debug = debug
self.logger = logging.getLogger(__name__)
if path:
    self.file = os.path.join(path,
    self.file.seek(0)
    self.fingerprints.update(self)

classmethod
def from_settings(cls, settings):
    debug = settings.getbool(
    return cls(job_dir(settings), debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

def request_fingerprint(self, request):
    return request_fingerprint(request)
```

# Comunicação com o usuário

## TUI - Text User Interface

- Texto:
  - Também composta por menus, ícones e janelas mas não são capazes de reproduzir figuras
  - Ex: Setup

```
self.file = None
self.fingerprints = self()
self.logdups = True
self.debug = debug
self.logger = logging.getLogger(__name__)
if path:
    self.file = os.path.join(path,
    self.file.seek(0)
    self.fingerprints.update(self)

classmethod
def from_settings(cls, settings):
    debug = settings.getbool
    return cls(job_dir(settings), debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

def request_fingerprint(self, request):
    return request_fingerprint(request)
```



# Comunicação com o usuário

## Exemplo de interface TUI

```
C:\PROGRAM FILES\MEDIA MACHINES
C:↓      Name      Size      Date
..      ▶UP--DIR◀  09.03.15
FLUX     ▶SUB-DIR◀  09.03.15
FluxStudio_2_1 ▶SUB-DIR◀  09.03.15
thumbnails ▶SUB-DIR◀  09.03.15

..      ▶UP--DIR◀  09.03.15  21:38

C:\PROGRAM FILES\MEDIA MACHINES>
1Left  2Right  3Name  4Exten  5Time  6Size  7Unsort  8Sync  9Print  10Split
```

C:\PROGRAM FILES\MEDIA MACHINES		
C:↓	Name	Date
..	▶UP--DIR◀	09.03.15
FLUX	▶SUB-DIR◀	09.03.15
FluxStudio_2_1	▶SUB-DIR◀	09.03.15
thumbnails	▶SUB-DIR◀	09.03.15
..	▶UP--DIR◀	09.03.15 21:38

### Info

The Norton Commander, Version 5.51  
1 July 1998

---

655 360 Bytes Memory  
560 480 Bytes Free  
2 147 155 968 total bytes on drive C:  
2 147 155 968 bytes free on drive C:  
0 files and 4 directories  
use 0 bytes in  
C:\PROGRAM FILES\MEDIA MACHINES

---

Volume Label : NO NAME  
Serial number: 3E53:10FE

---

No "dirinfo" file in this directory

```
self.file = None
self.fingerprints = self()
self.logdups = True
self.debug = debug
self.logger = logging.getLogger(__name__)
if path:
    self.file = os.path.join(path,
    self.file.seek(0)
    self.fingerprints.update(self)

classmethod
def from_settings(cls, settings):
    debug = settings.getbool
    return cls(job_dir(settings), debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

def request_fingerprint(self, request):
    return request_fingerprint(request)
```

# Comunicação com o usuário

## CUI/CLI - Command Line Interface

- Linha de comando (Command-line User Interface) ou **SHELL**
  - Funciona basicamente com digitação de comandos
    - Ex: MS-DOS, Terminal do Linux



```
self.file = None
self.fingerprints = self()
self.logdupes = True
self.debug = debug
self.logger = logging.getLogger(__name__)
if path:
    self.file = open(os.path.join(path,
    self.file.seek(0)
    self.fingerprints.update(self.logger)

@classmethod
def from_settings(cls, settings):
    debug = settings.get('debug')
    return cls(job_dir(settings), debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

def request_fingerprint(self, request):
    return request_fingerprint(request)
```

# Comunicação com o usuário

## Exemplo de interface CUI/CLI

### CMD no Windows

```
Current date is Tue 1-01-1980
Enter new date:
Current time is 7:48:27.13
Enter new time:

The IBM Personal Computer DOS
Version 1.10 (C)Copyright IBM Corp 1981, 1982

A>dir/w
COMMAND  COM      FORMAT  COM      CHKDSK   COM      SYS       COM      DISKCOPY  COM
DISKCOMP  COM      COMP     COM      EXE2BIN  EXE      MODE      COM      EDLIN     COM
DEBUG     COM      LINK     EXE      BASIC    COM      BASICA    COM      ART       BAS
SAMPLES   BAS      MORTGAGE BAS      COLORBAR BAS      CALENDAR  BAS      MUSIC     BAS
DONKEY     BAS      CIRCLE   BAS      PIECHART BAS      SPACE     BAS      BALL      BAS
COMM      BAS

26 File(s)
A>dir command.com
COMMAND  COM      4959   5-07-82  12:00p
1 File(s)
A>
```

### Terminal no Linux

```
override@Atul-HP: ~
override@Atul-HP:~$ ls -l
total 212
drwxrwxr-x  5 override override 4096 May 19 03:45 acadenv
drwxrwxr-x  4 override override 4096 May 27 18:20 acadview_demo
drwxrwxr-x 12 override override 4096 May  3 15:14 anaconda3
drwxr-xr-x  6 override override 4096 May 31 16:49 Desktop
drwxr-xr-x  2 override override 4096 Oct 21 2016 Documents
drwxr-xr-x  7 override override 4096 Jun  1 13:09 Downloads
-rw-r--r--  1 override override 8980 Aug  8 2016 examples.desktop
-rw-rw-r--  1 override override 45005 May 28 01:40 hs_err_pid1971.log
-rw-rw-r--  1 override override 45147 Jun  1 03:24 hs_err_pid2006.log
drwxr-xr-x  2 override override 4096 Mar  2 18:22 Music
drwxrwxr-x 21 override override 4096 Dec 25 00:13 Mydata
drwxrwxr-x  2 override override 4096 Sep 20 2016 newbin
drwxrwxr-x  5 override override 4096 Dec 20 22:44 nltk_data
drwxr-xr-x  4 override override 4096 May 31 20:46 Pictures
drwxr-xr-x  2 override override 4096 Aug  8 2016 Public
drwxrwxr-x  2 override override 4096 May 31 19:49 scripts
drwxr-xr-x  2 override override 4096 Aug  8 2016 Templates
drwxrwxr-x  2 override override 4096 Feb 14 11:22 test
drwxr-xr-x  2 override override 4096 Mar 11 13:27 Videos
drwxrwxr-x  2 override override 4096 Sep  1 2016 xdm-helper
override@Atul-HP:~$
```

```
self.file = None
self.fingerprints = self()
self.logdups = True
self.debug = debug
self.logger = logging.getLogger(__name__)
if path:
    self.file = os.path.join(path,
    self.file.seek(0)
    self.fingerprints.update(self)

classmethod
def from_settings(cls, settings):
    debug = settings.getbool(
    return cls(job_dir(settings), debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

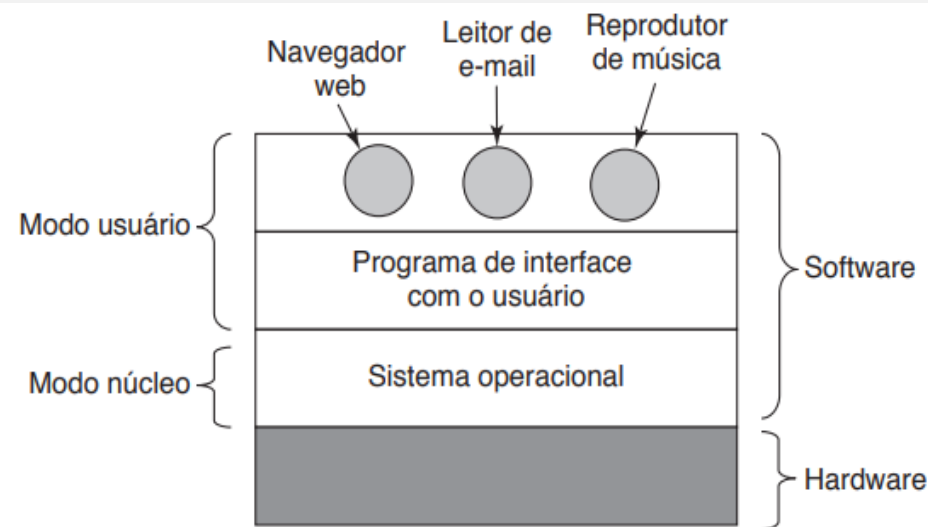
def request_fingerprint(self, request):
    return request_fingerprint(request)
```

# Relação das Interfaces com o SO

O programa de interface com o usuário, **shell ou GUI**, é o **nível mais inferior de software de modo usuário**

Essas interfaces permitem que ele inicie outros programas, como um navegador web, leitor de e-mail, ou reproduutor de música

Um exemplo: estando no CMD do Windows, ao digitar "explorer", será aberto o Windows Explorer no modo GUI.



```
self.file = None
self.fingerprints = self()
self.logdups = True
self.debug = debug
self.logger = logging.getLogger(__name__)
if path:
    self.file = os.path.join(path,
    self.file.seek(0)
    self.fingerprints.update(self)

classmethod
def from_settings(cls, settings):
    debug = settings.getbool(
    return cls(job_dir(settings), debug)

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    self.fingerprints.add(fp)
    if self.file:
        self.file.write(fp + os.linesep)

def request_fingerprint(self, request):
    return request_fingerprint(request)
```