Disciplina de Sistemas Operacionais (Aula 2)

Introdução ao Shell Script

Apresentação

Francisco Nauber Bernardo Gois

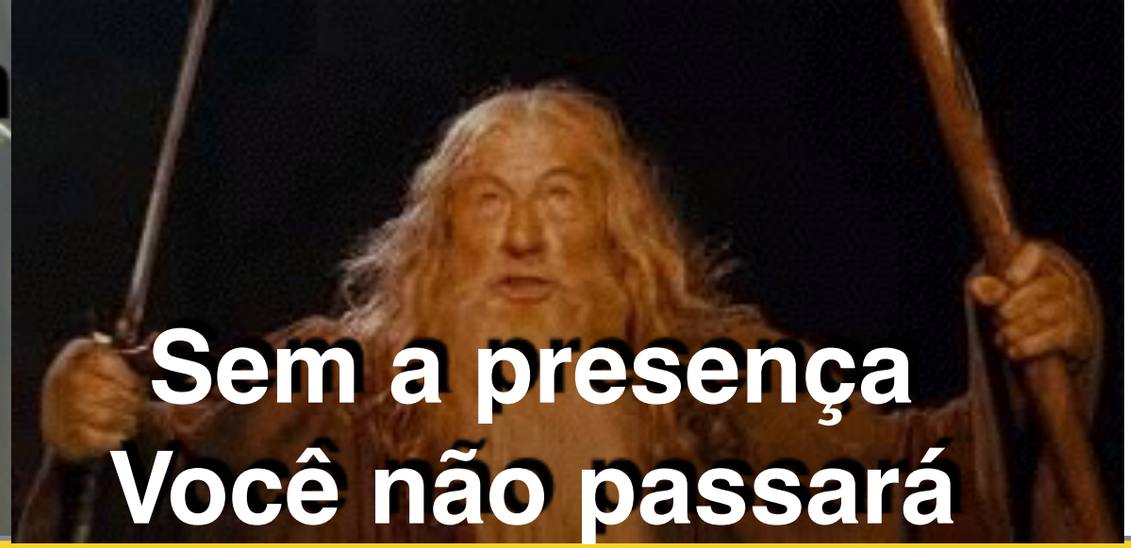
Analista aprendizado de máquina
no Serviço Federal de Processamento
de Dados

Doutorando em Informática Aplicada

Mestre em Informática Aplicada

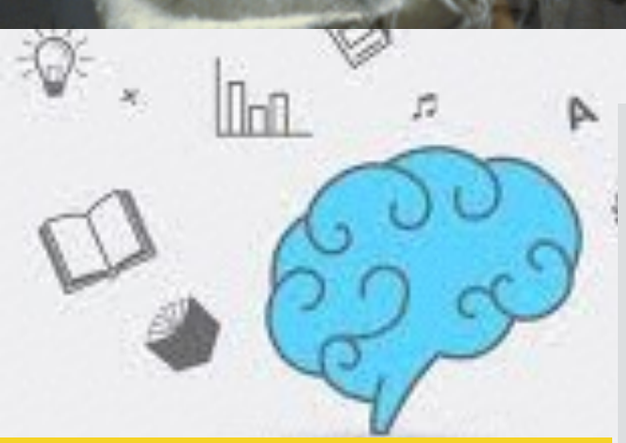
Especialista em desenvolvimento WEB

Dúvidas: naubergois@gmail.com



Para melhor desempenho na aula

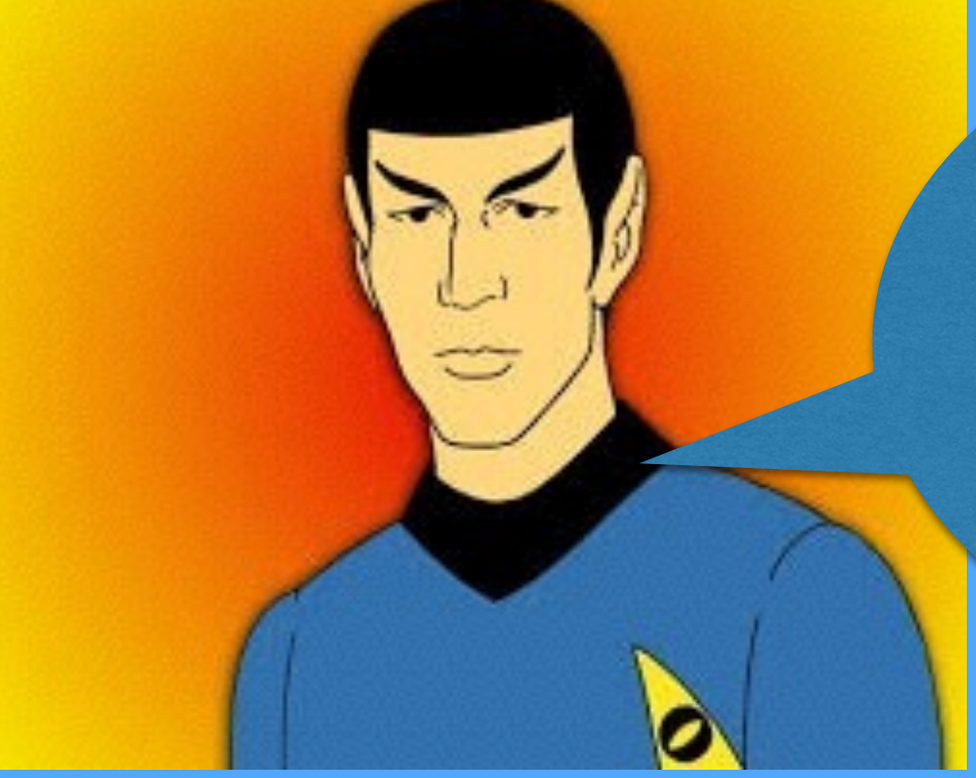
Cuidado com o Horário



Procure não conversar durante a aula

Buscar aprendizado ao invés de pontos

Não teremos pontuação fora dos trabalhos e provas da disciplina

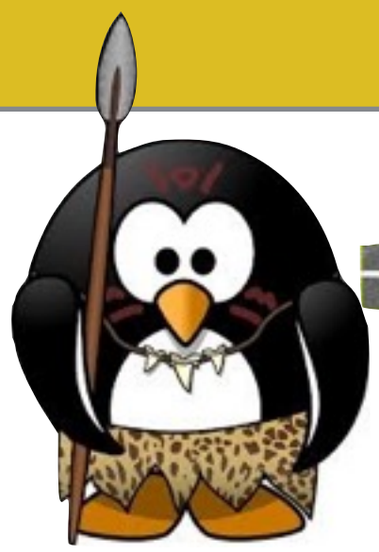


OS trabalhos deveram ser entregues uma semana antes da prova

Um cadeira longa e prospera

Não teremos pontos após a prova não adianta pedir

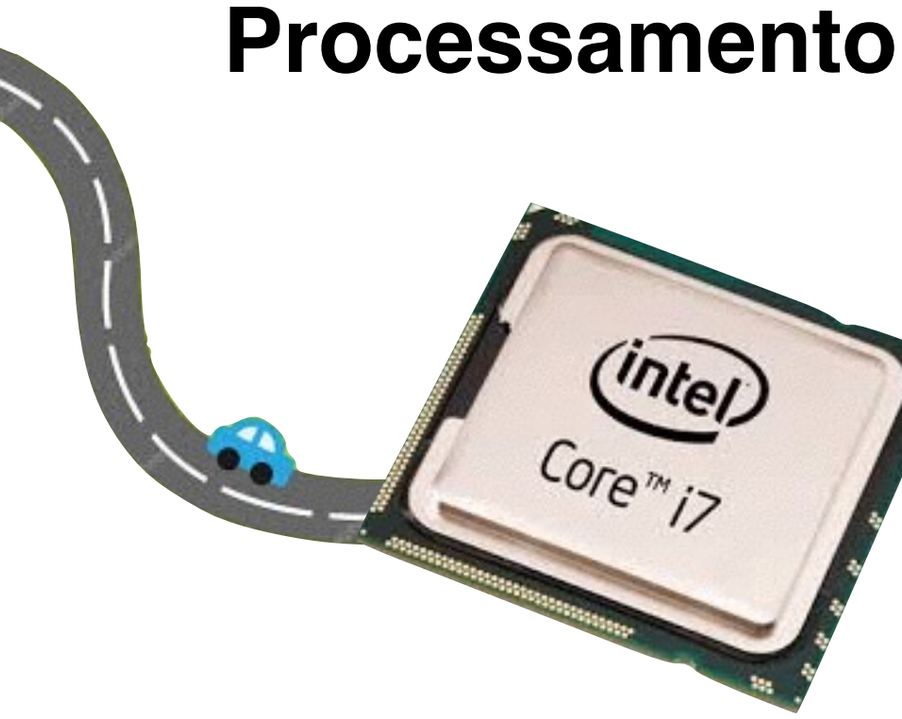
Cronograma da Disciplina



Introdução ao SO

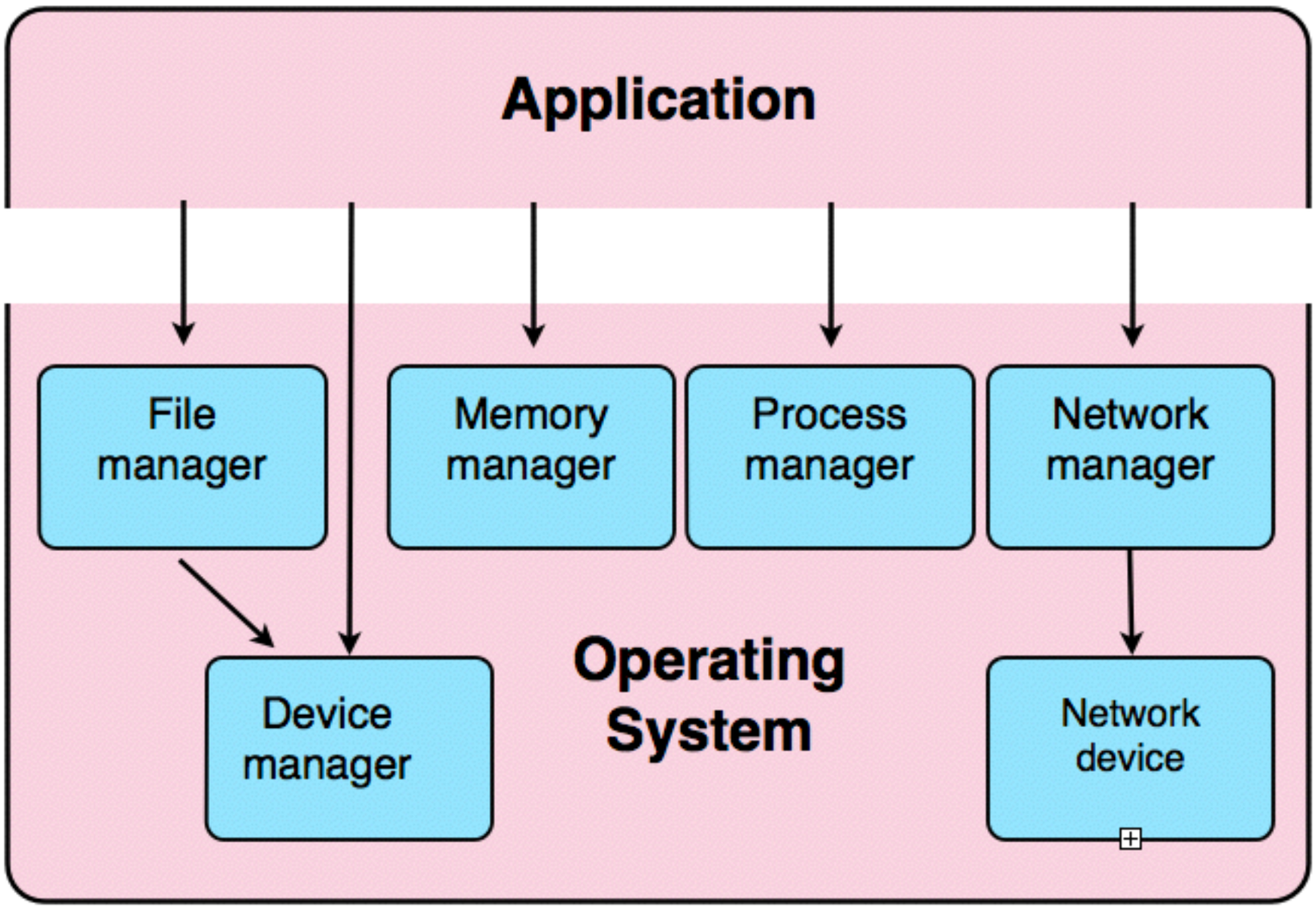
Introdução ao Shell Script

Gerencia de Processamento



Dúvidas: naubergois@gmail.com

Gerências de recursos dos Sistemas Operacionais



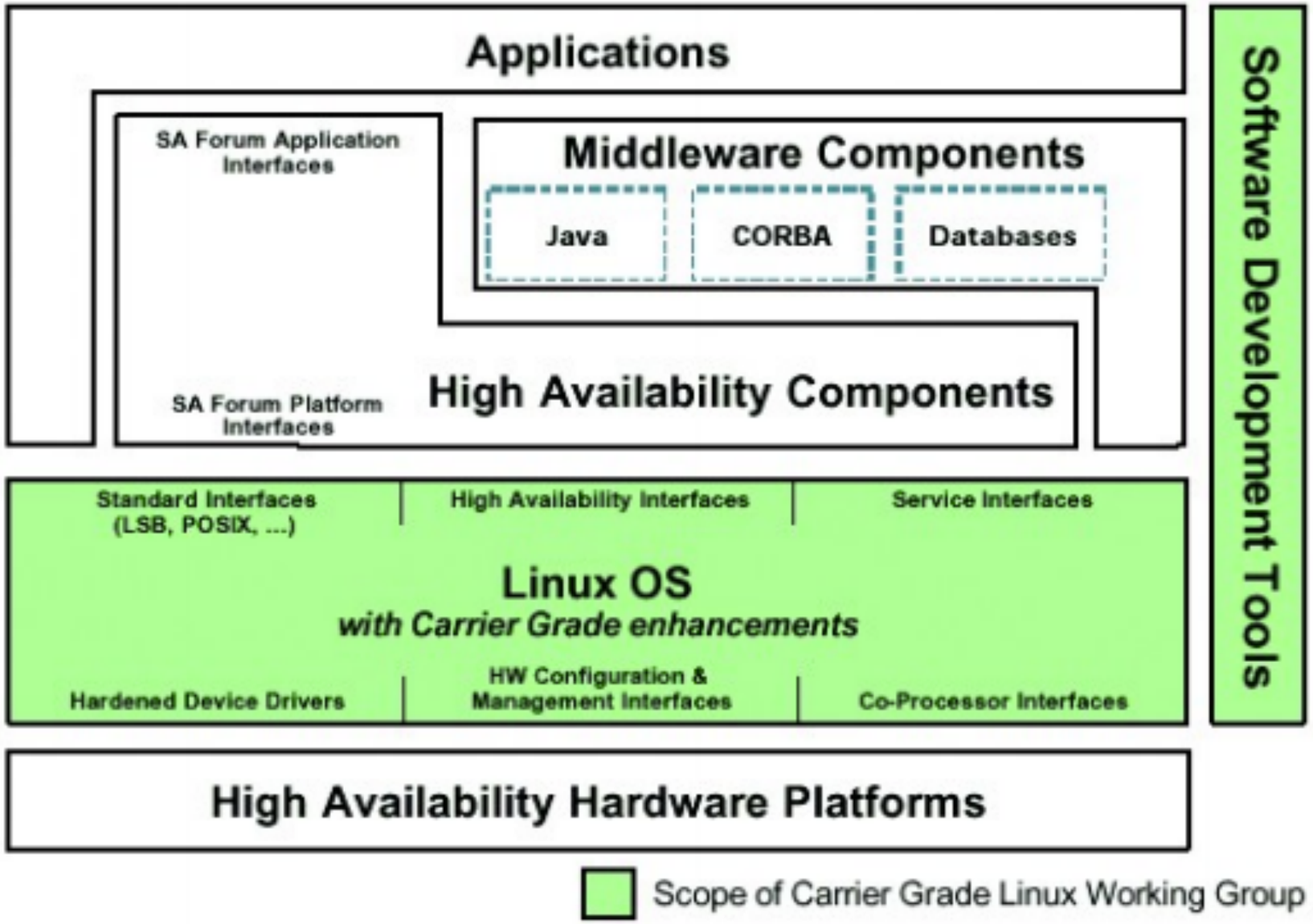
O que vimos na aula passada

- Introdução ao SO
- Introdução ao Docker

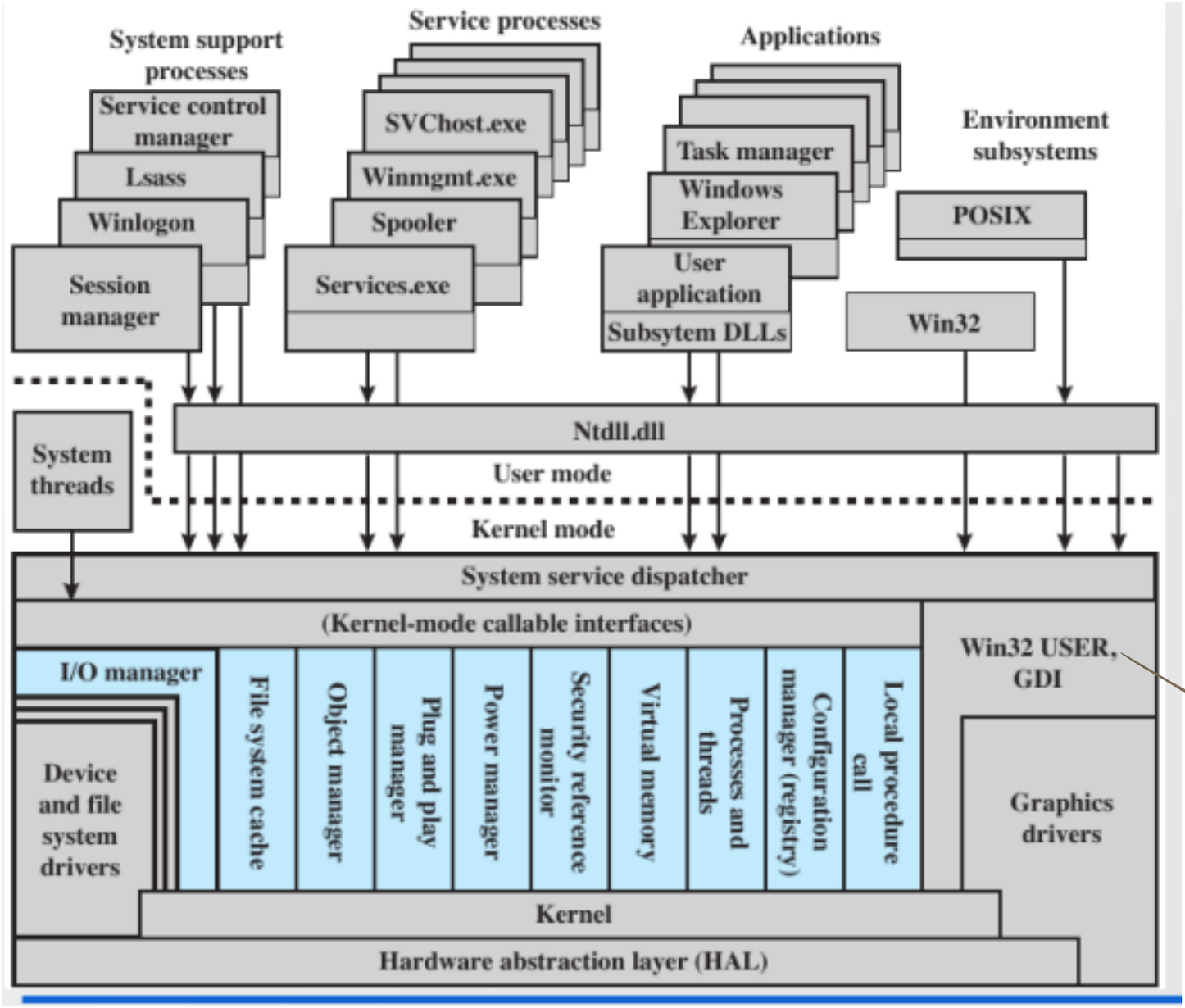
Objetivo da Aula

- Introdução ao Shell Script
- Gerência de Processamento
- Introdução ao Docker

Estrutura dos Sistemas Operacionais Linux



Estrutura dos Sistemas Operacionais Windows



makeuseof.com

Unix/Linux Command Reference

File commands

ls	Directory listing
ls -al	Formatted listing with hidden files
cd dir	Change directory to dir
cd	Change to home
pwd	Show current directory
mkdir dir	Create a directory dir
rm file	Delete file
rm -r dir	Delete directory dir
rm -f file	Force remove file
rm -rf dir	For remove directory dir
cp file1 file2	Copy file1 to file2
cp -r dir1 dir2	Copy dir1 to dir2; create dir2 if it doesn't exit
mv file1 file2	Rename or move file1 to file2. If filed 2 is an existing directory, moves file1 into directory file2
ln -s file link	Create symbolic link link to file
touch file	Create or update file
cat > file	Places standard input into file
more file	Output the contents of file
head file	Output the first 10 lines of file
tail file	Output the last 10 lines of file
tail -f file	Output the contents of file as it grows, starting with the last 10 lines

Process Management

ps	display all currently active processes
top	display all running processes
kill pid	kill process id pid
killall proc	kill all processes named proc *
bg	lists stopped or background jobs; resume a stopped job in the background
fg	Brings the most recent job to the foreground
fg a	brings job a to the foreground

File Permissions

chmod octal file	change the permissions of file to octal, which can be found separately for user, group, and world by adding: • 4 – read (r) • 2 – write (w) • 1 – execute (x) **Examples:** ***chmod 777*** *– read, write, execute for all* ***chmod 755*** *– rwx for owner, rx for group and world. For more options, see* **man chmod.**

SSH

ssh user@host	connect to host as user
ssh -p port user@host	connect to host on port port as user
ssh-copy-id user@host	add your key to host for user to enable a keyed or passwordless login

Searching

grep pattern files	search for pattern in files
grep -r pattern dir	search recursively for pattern in dir
command | grep pattern	search for pattern in the output of command
locate file	find all instances of file

System Info

date	show the current date and time
cal	show this month's calendar
uptime	show current uptime
w	display who is online
whoami	who you are logged in as
finger user	display information about user
uname -a	show kernel information
cat /proc /cpuinfo	cpu information
cat /proc /meminfo	memory information
man command	show the manual for command
df	show disk usage
du	show directory space usage
free	show memory and swap usage
whereis app	show possible locations of app
which app	show which app will be run by default

Compression

tar cf file.tar files	create a tar named file.tar containing files
tar xf file.tar	extract the files from file.tar
tar czf file.tar.gz files	create a tar with Gzip compression
tar xzf file.tar.gz	extract a tar using Gzip
tar cjf file.tar.bz2	create a tar with Bzip2 compression
tar xjf file.tar.bz2	extract a tar using Bzip2
gzip file	compresses file and renames it to file.gz
gzip -d file.gz	decompresses file.gz back to file

Network

ping host	ping host and output results
whois domain	get whois information for domain
dig domain	get DNS information for domain
dig -x host	reverse lookup host
wget file	download file
wget -c file	continue a stopped download

Installation

Install from source:	
./configure	
make	
make install	
dpkg -i pkg.deb	install a package (Debian)
rpm -Uvh pkg.rpm	install a package (RPM)

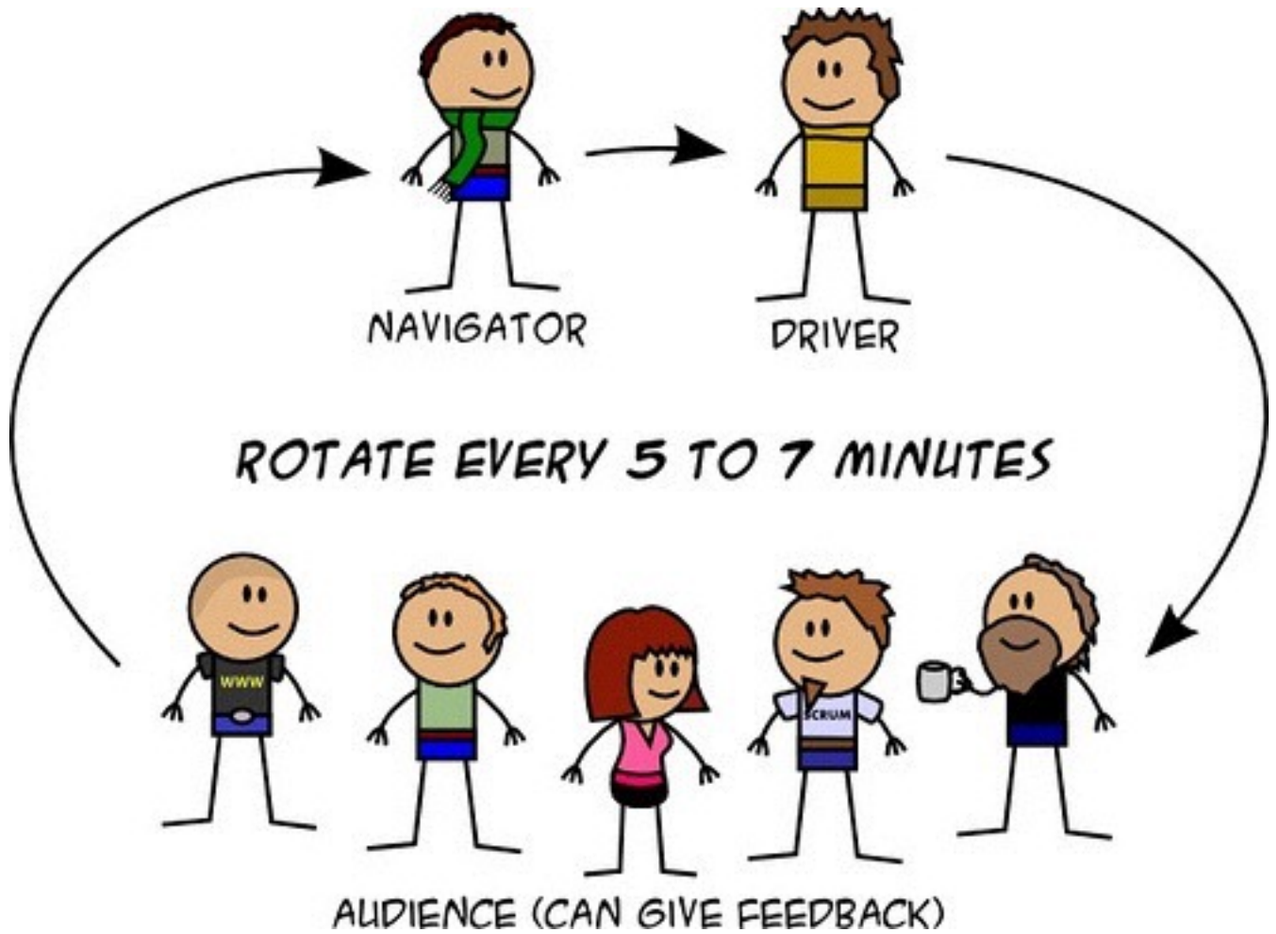
Shortcuts

Ctrl+C	halts the current command
Ctrl+Z	stops the current command, resume with fg in the foreground or bg in the background
Ctrl+D	log out of current session, similar to exit
Ctrl+W	erases one word in the current line
Ctrl+U	erases the whole line
Ctrl+R	type to bring up a recent command
!!	repeats the last command
exit	log out of current session
*	use with extreme caution

Introdução ao Shell Script

touch exemplo1.sh

Coding Dojo



CODING DOJO

Introdução ao Shell Script

```
#!/bin/bash
  echo "Seu nome de usuário é:"
  whoami
  echo "Info de hora atual e tempo
que o computador está ligado:"
   uptime
   echo "O script está executando
 do diretório:"
   pwd
```

CODING DOJO

Introdução ao Shell Script

```
#!/bin/bash
#Este é um comentário
#Este é outro comentário
echo "Este script contém comentários."
```

CODING DOJO

Declarando variáveis

```
#!/bin/bash
site=www.google.com
meu_numero_favorito=13
_cidade=”Porto Alegre”
  echo “Um ótimo site para você aprender a programar e se manter atualizado é: $site”
  echo “Meu número favorito é: $meu_numero_favorito”
  echo “Minha cidade natal é: $_cidade”
```

CODING DOJO

Realizando Leitura de Dados

```
#!/bin/bash
 echo “Digite um número qualquer:”
 read numero;
 if [ “$numero” -gt 20 ];
 then
   echo “Este número é maior que 20!”
 fi
```

CODING DOJO

Condicionais

string1 = string2: string1 e string2 são idênticas;
string1 != string2: string1 e string2 são diferentes;
inteiro1 -eq inteiro2: inteiro1 possui o mesmo valor que inteiro2;
inteiro1 -ne inteiro2: inteiro1 não possui o mesmo valor que inteiro2;
inteiro1 -gt inteiro2: inteiro1 é maior que inteiro2;
inteiro1 -ge inteiro2: inteiro1 é maior ou igual a inteiro2;
inteiro1 -lt inteiro2: inteiro1 é menor que inteiro2;
inteiro1 -le inteiro2: inteiro1 é menor ou igual a inteiro2;

Usando Condicionais

```
#!/bin/bash
echo "Selecione uma opção:"
echo "1 - Exibir data e hora do sistema"
echo "2 - Exibir o resultado da divisão 10/2"
echo "3 - Exibir uma mensagem”
read opcao;
if [ $opcao == "1" ];
```

CODING DOJO

Usando Condicionais

```
then
  data=$(date +"%T, %d/%m/%y, %A")
  echo "$data"
elif [ $opcao == "2" ];
```

CODING DOJO

Usando Condicionais

```
then
   result=$((10/2))
   echo "divisao de 10/2 = $result"
elif [ $opcao == "3" ];
 then
 echo "Informe o seu nome:"
 read nome;
 echo "Bem-vindo ao mundo do shell script, $nome!"
fi
```

CODING DOJO

Usando Condicionais

```
then
   result=$((10/2))
   echo "divisao de 10/2 = $result"
elif [ $opcao == "3" ];
 then
 echo "Informe o seu nome:"
 read nome;
 echo "Bem-vindo ao mundo do shell script, $nome!"
fi
```

CODING DOJO

Processos

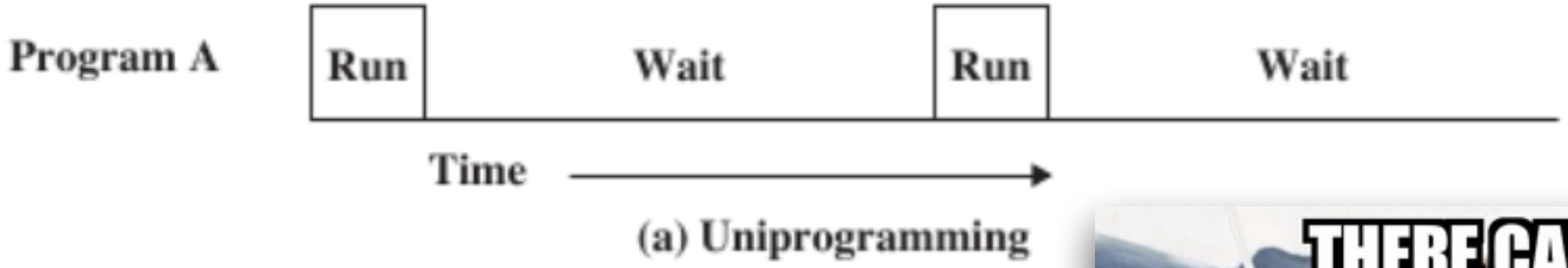
- Processo: chave do SO;
  - Caracterizado por programas em execução;
  - Cada processo possui:
    - Programa (instruções que serão executadas);
    - Um espaço de endereço de memória (max e min);
    - Contextos de hardware: informações de registradores;
    - Contextos de software: atributos;
- O Sistema Operacional gerencia todos os processos → bloco de controle de processo;

Processos

- **Processo não é o mesmo que programa !**
- Programa é como uma receita, com instruções para se resolver um problema
- Processo é como alguém seguindo a receita
- Processo precisa de recursos computacionais para trabalhar: memória, arquivos, tempo de processador, ...
- Programa não trabalha: fica passivamente guardado em um arquivo

Sistema Operacional Monoprogramado

SO pode ser monoprogramado: apenas um processo executa por vez (começo ao fim):



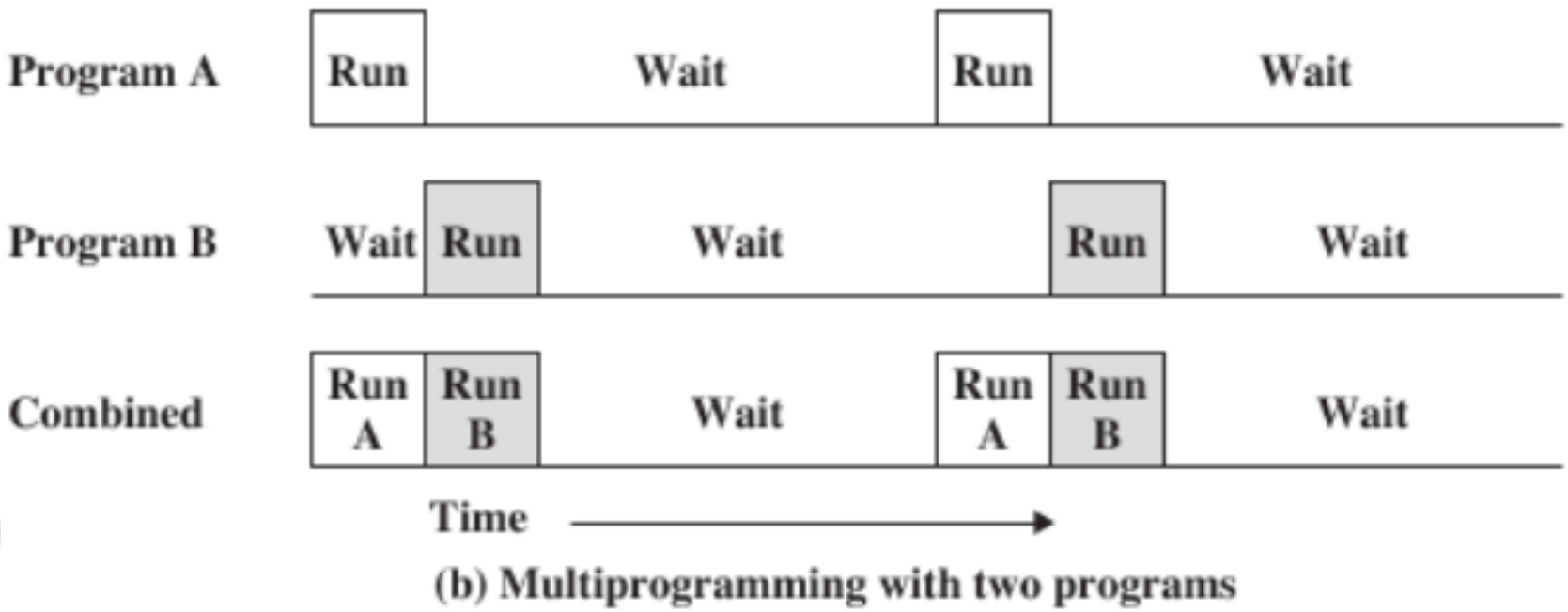
(a) Uniprogramming

Dúvidas: naubergois@gmail.com

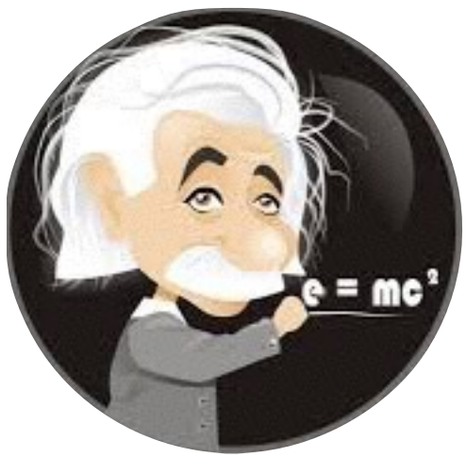
Sistema Operacional Monoprogramado

Sistema Operacional Multiprogramados

... ou multiprogramado (mais de um processo pode executar ao mesmo tempo):



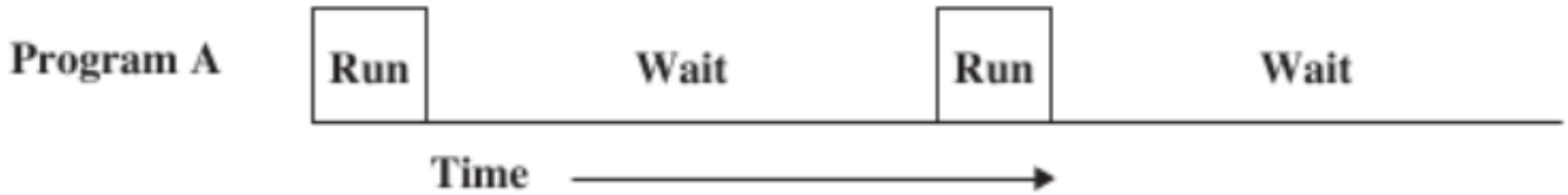
(b) Multiprogramming with two programs



Porque existe multiprogramação

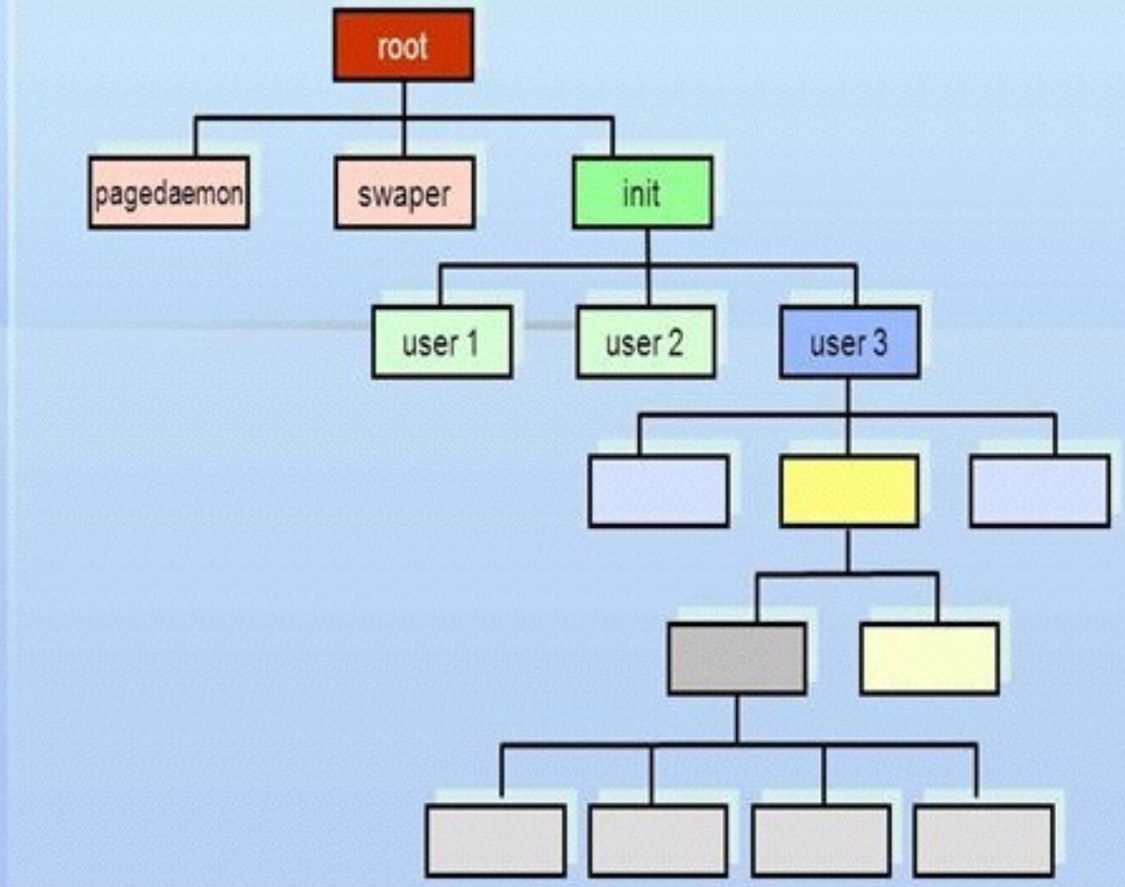
Por que existe multiprogramação ?

- Processos alternam entre uso do processador ou espera por E/S:

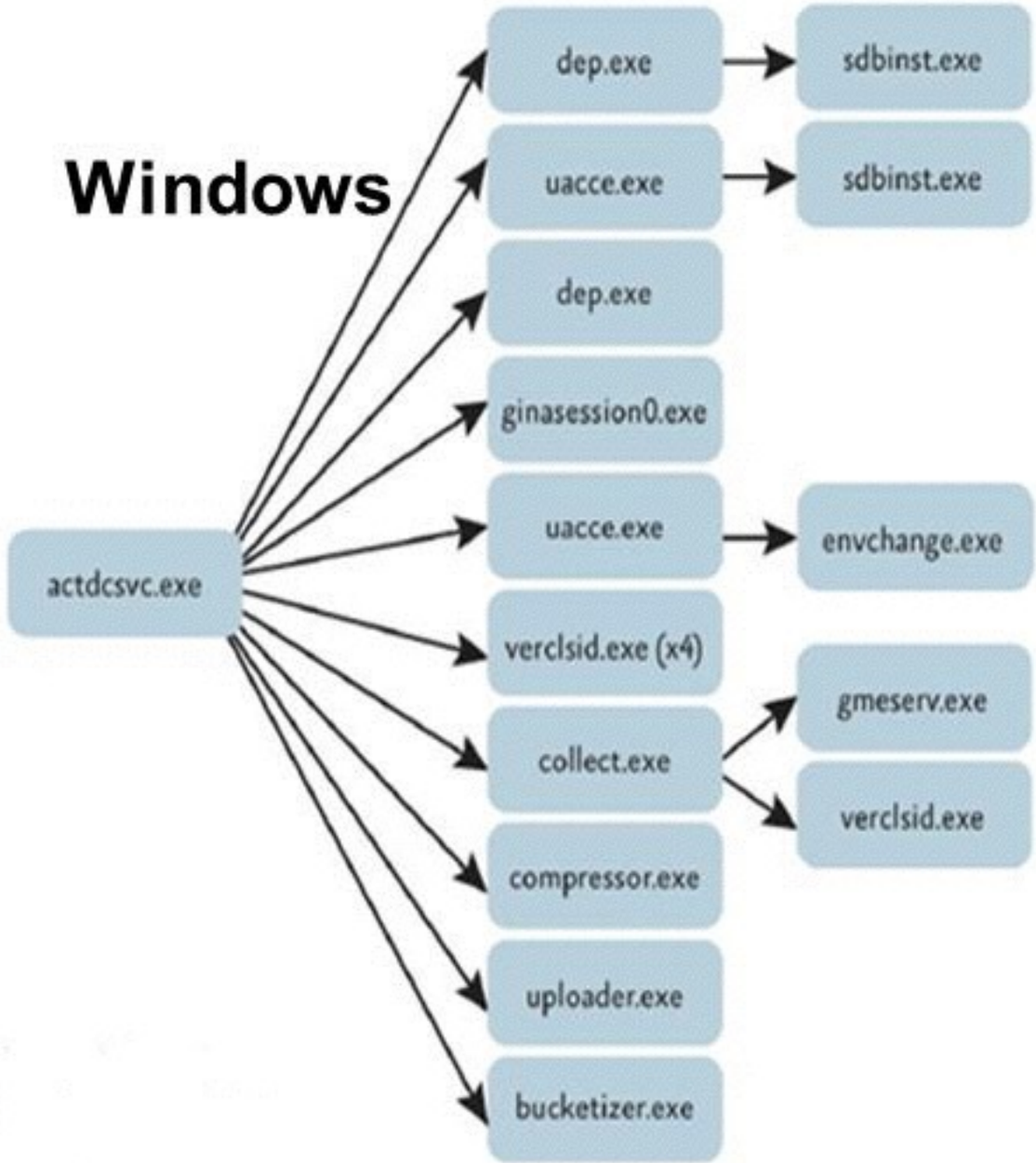


Árvore de Processos

Árvore de Processos: SO Unix Típico



Windows



Porque existe multiprogramação

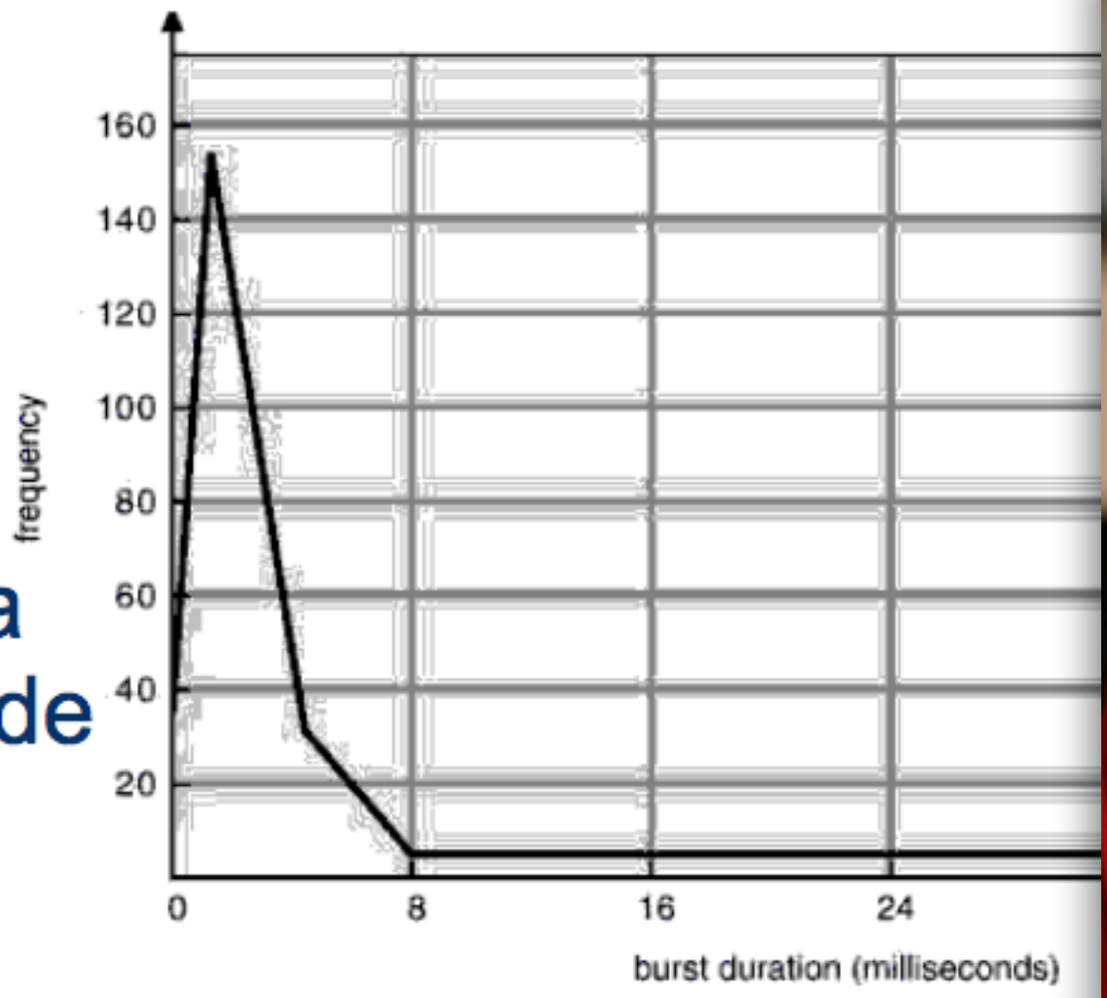
Por que existe multiprogramação ?

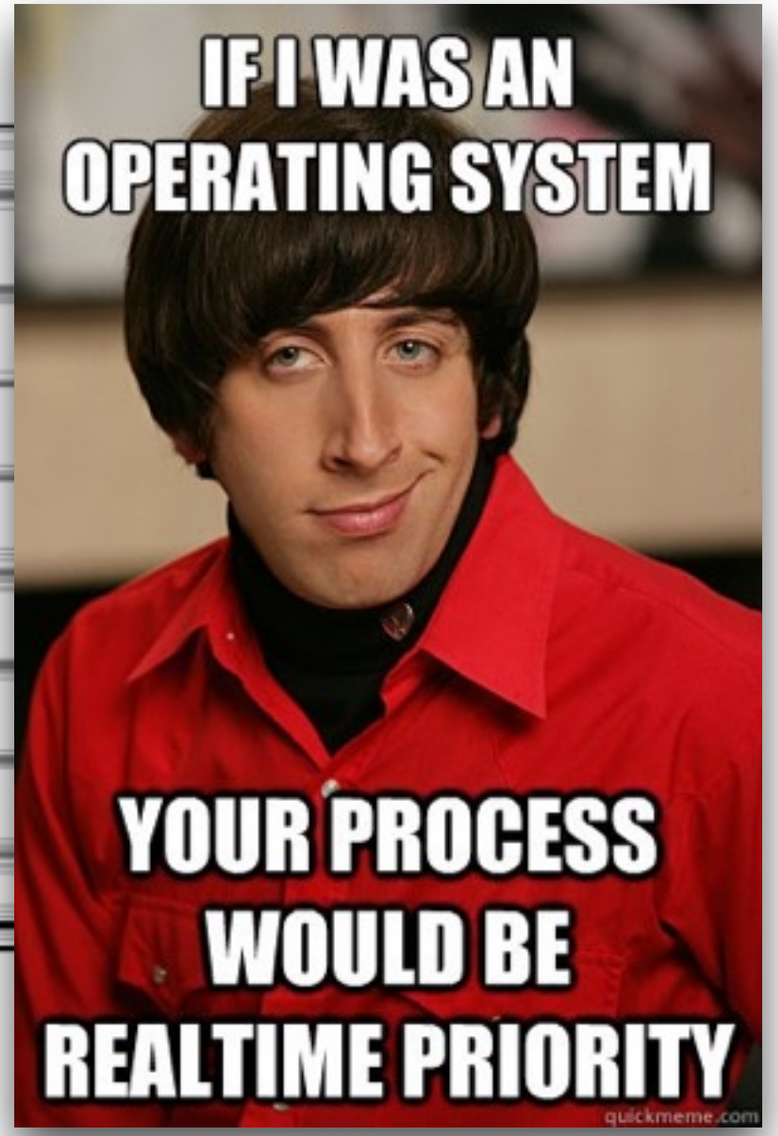
- Processos podem ser classificados quanto ao uso do processador:
  - **Processos I/O bound**
    - Passam mais tempo em estado de espera que usando de fato o processador
  - **Processos CPU bound**
    - Passam a maior parte do tempo usando de fato o processador

Porque existe multiprogramação

Por que existe multiprogramação ?

Um histograma dos tempos de execução





Porque existe multiprogramação

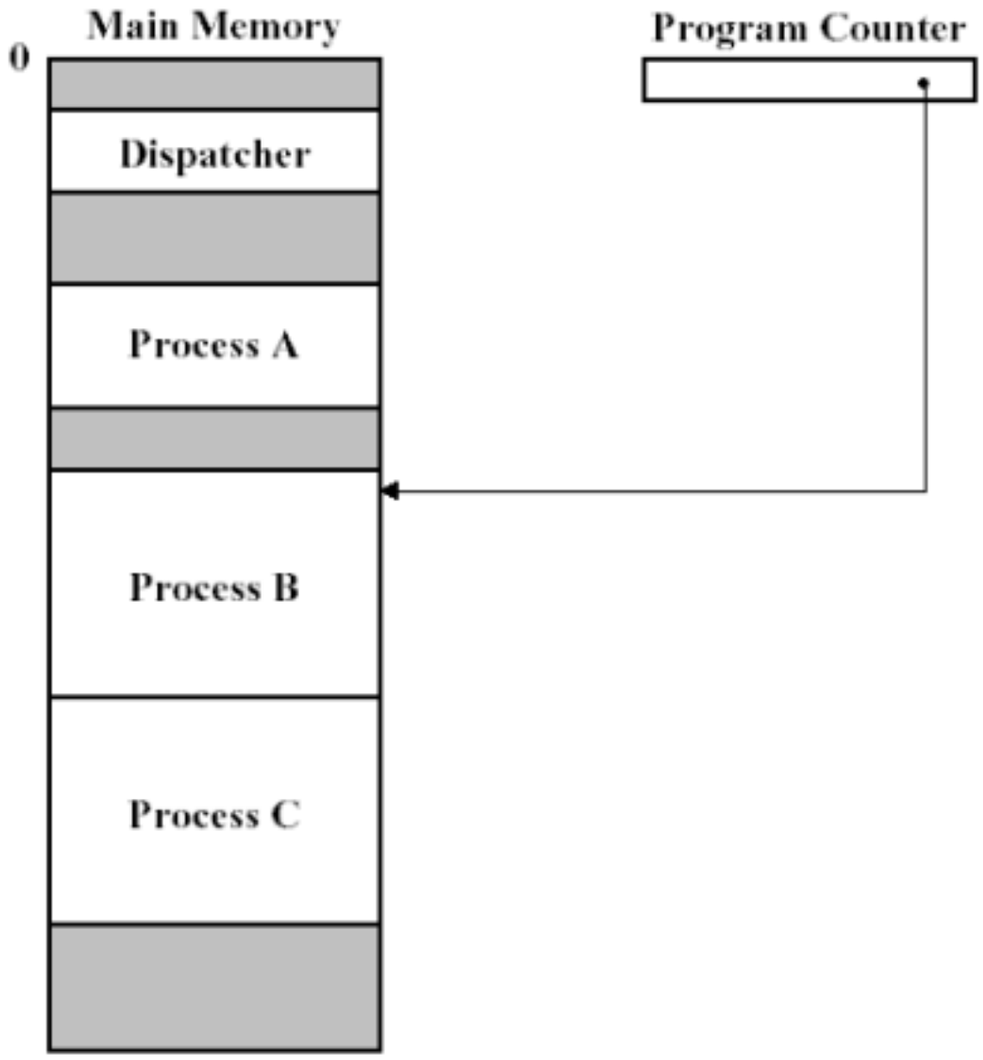
Por que existe multiprogramação ?

- Um processo típico passa mais tempo em espera que processando
- Pode-se aproveitar os tempos de espera para executar outros processos
- Assim se aproveita melhor o tempo de processador disponível !

Dúvidas: naubergois@gmail.com

Multiprogramação

- **Multiprogramação**
  - Capacidade do sistema operacional de manter vários processos ativos ao mesmo tempo
  - *Timesharing*: ilusão de que os processos executam ao mesmo tempo



Multiprogramação no Linux - Comando Top

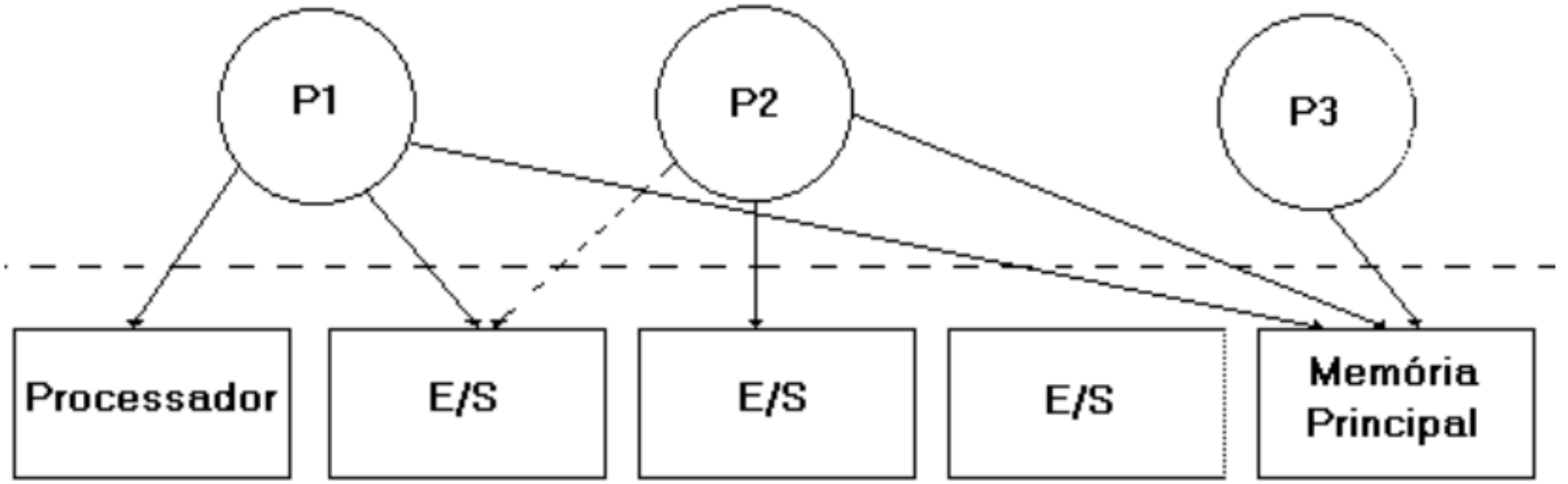
- **Multiprogramação no Linux**
  - Listagem dos processos que mais usam processador

```
Tasks: 128 total,   3 running, 123 sleeping,   0 stopped,   2 zombie
Cpu(s): 15.5%us,  1.3%sy,  0.0%ni, 83.2%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1025208k total,   972208k used,    53000k free,    26016k buffers
Swap:  1542200k total,    17476k used,  1524724k free,   407096k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 6121 sobral    20   0  239m 111m  27m R   27 11.1  13:12.65 firefox
 2425 root      20   0  400m  56m 3388 S    4  5.6   3:23.20 Xorg
 4450 sobral    20   0 45740 4664 2884 S    1  0.5   1:00.06 xmms
 6471 sobral    20   0  147m  27m  12m S    1  2.7   0:42.03 knotify4
 4227 sobral    20   0 35684  10m 8064 S    1  1.0   0:12.20 kdesktop
 4447 sobral    20   0 31720  11m 8520 R    1  1.2   0:02.74 konsole
 4501 sobral    20   0 79048  27m  11m S    0  2.8   0:15.94 skype
 4716 sobral    20   0  332m 116m  54m S    0 11.6   2:17.00 soffice.bin
```

Processos e recursos do computador

Processos usam recursos do computador



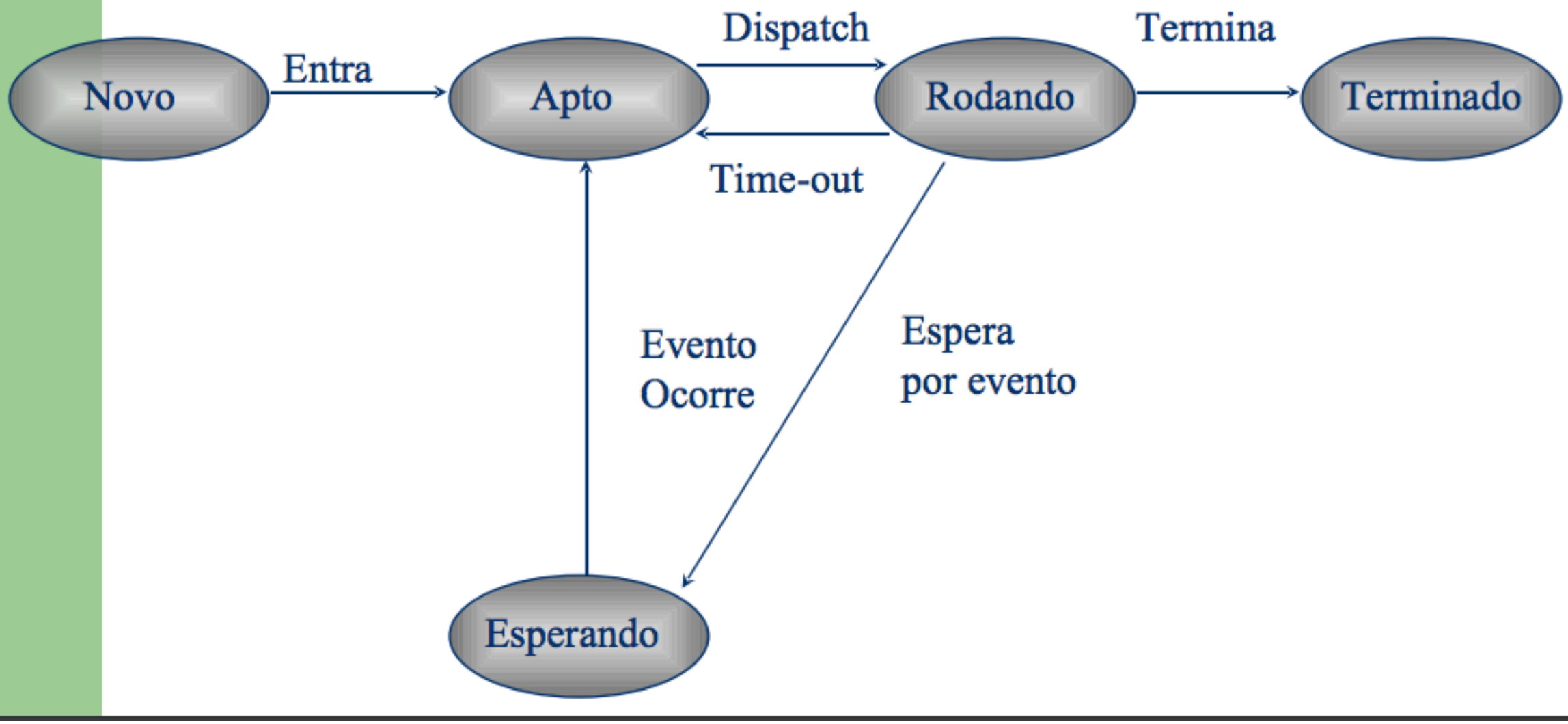
Processos e recursos do computador

Como o SO controla os processos existentes ?

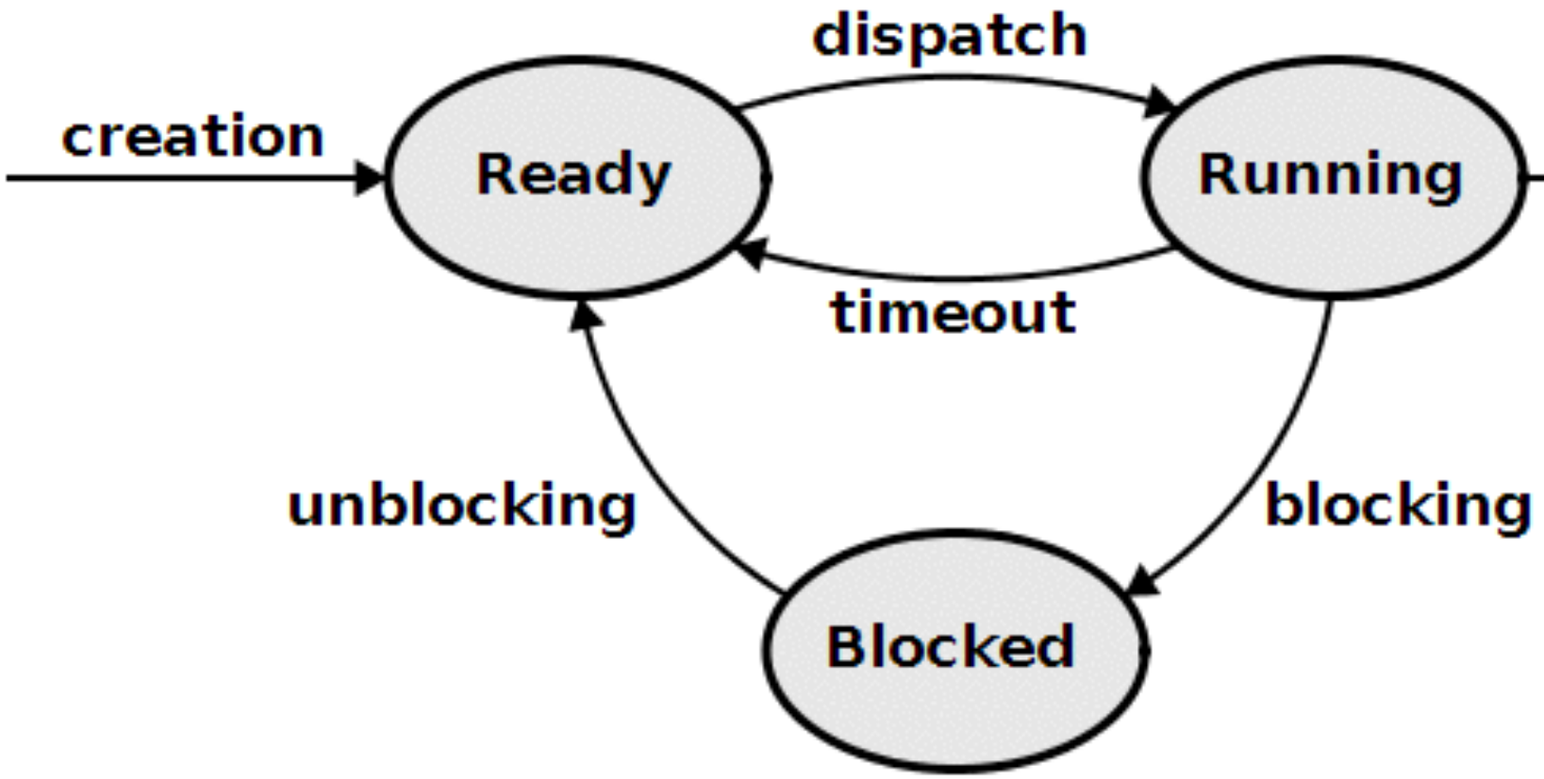
- SO precisa decidir que processo deve usar o processador
- Somente processos aptos a executar podem usar o processador
- **Processo apto**: *que não está esperando por E/S*

Ciclo de vida de um processo

Modelo de cinco estados

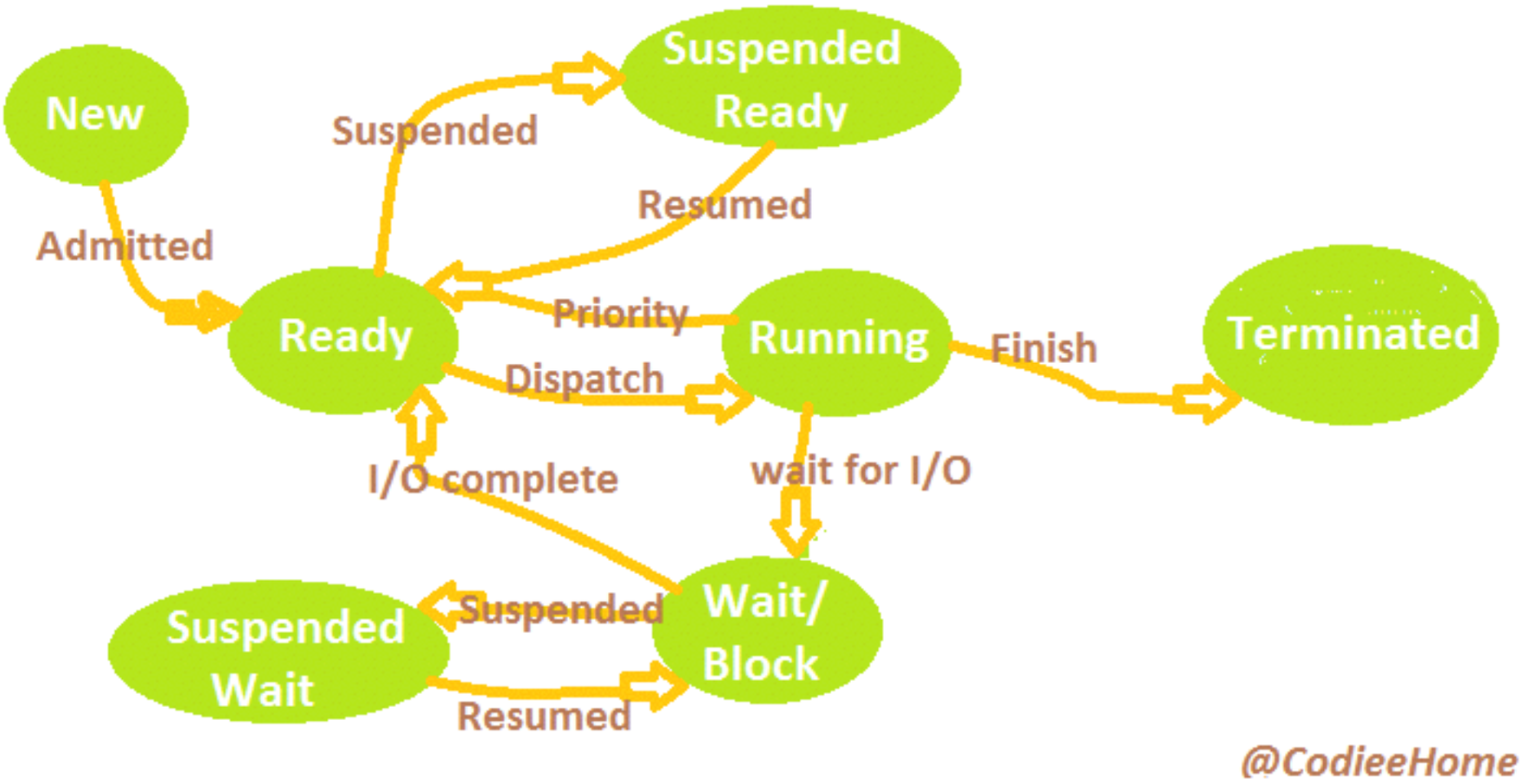


Ciclo de vida de um processo



Ciclo de vida de um processo

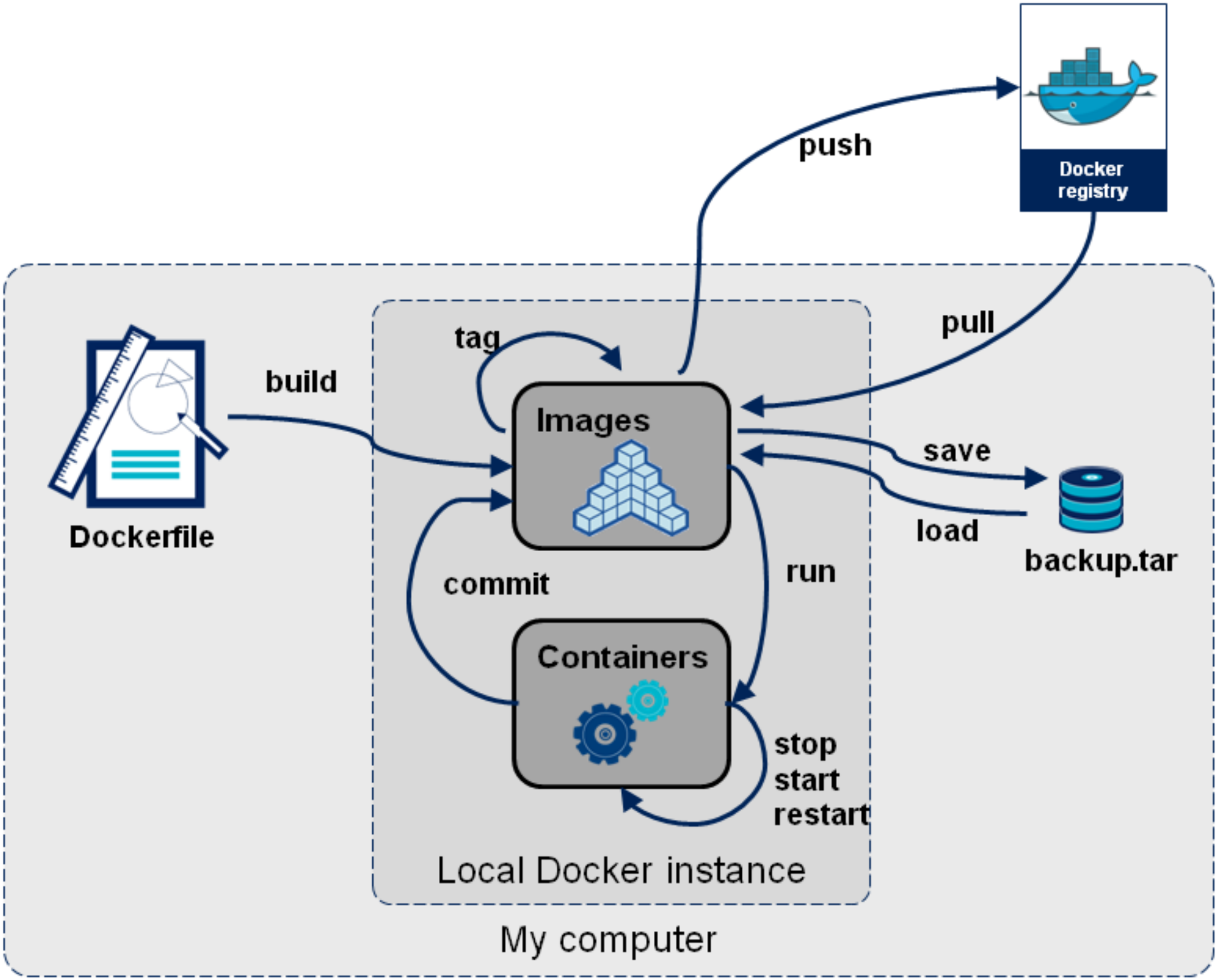
Life Cycle Of a Process-



Introdução ao Docker

Docker não é um sistema de virtualização tradicional.

Enquanto em um ambiente de virtualização tradicional nós temos um S.O. completo e isolado, dentro do Docker nós temos recursos isolados que utilizando bibliotecas de kernel em comum (entre host e container)


Docker registry
push
pull
tag
build
Images
save
load
Dockerfile
backup.tar
run
commit
Containers
stop
start
restart
Local Docker instance
My computer

Introdução ao Docker

docker push [OPTIONS] NAME[:TAG]

docker push registry-host:5000/myadmin/rhel-httpd

DockerFile

```
FROM ImageName
# directive=value
```

```
FROM kstaken/apache2

MAINTAINER Kimbro Staken version: 0.
```

```
FROM ubuntu:12.04

MAINTAINER Kimbro Staken ve
```

Conhecimentos Adquiridos

- Aprender conceitos de gerenciamento de Processos
- Aprender conceitos inicias sobre o uso de shell script

Francisco Nauber Bernardo Gois
Email: naubergois@gmail.com