UFC CAMPUS QUIXADÁ

Gerenciamento de Processos

Prof. Michel Sales Bonfim

Disciplina: Administração de Sistemas Operacionais Linux

Processos

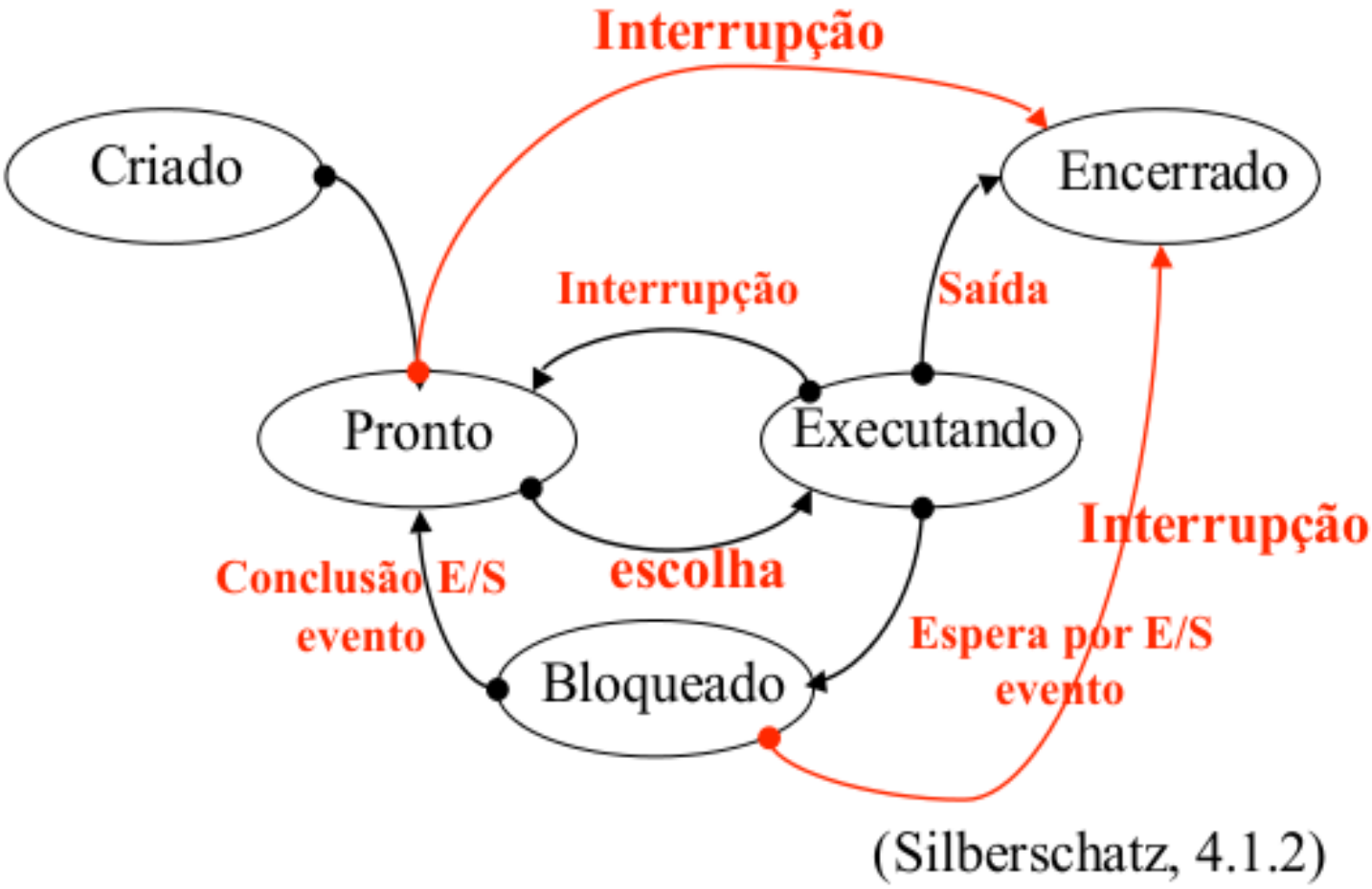
Processo

- Um **programa** é:
  - Uma sequência finita de instruções;
  - Uma entidade passiva (que não se altera com o passar do tempo);
  - Armazenado em disco.
- Um **processo** é:
  - Uma **abstração** que representa um programa em execução;
  - Uma entidade dinâmica: seu estado se altera conforme for executando;
  - Armazenado em memória RAM.
- Pode-se encontrar mais de um processo instanciando um único programa.

Ciclo de Vida do Processo

- Processos **nascem**
  - No momento de sua criação (via chamada de sistema)
- Processos **vivem**
  - Executam na CPU, liberam a CPU (E/S)...
  - Estados: Pronto, Executando, Bloqueado.
  - Executam:
    - Programas dos usuários
    - Programas do sistema (daemons)
- Processos **morrem**
  - Ou porque terminaram sua execução
  - Ou porque um outro processo os matou:
    - Erro, acesso não-autorizado, falha

Ciclo de Vida de um Processo

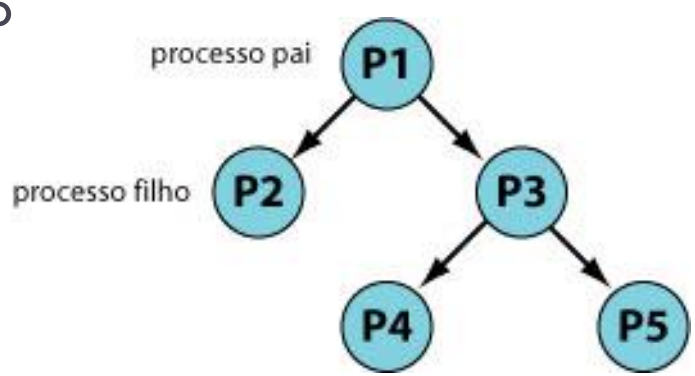


(Silberschatz, 4.1.2)

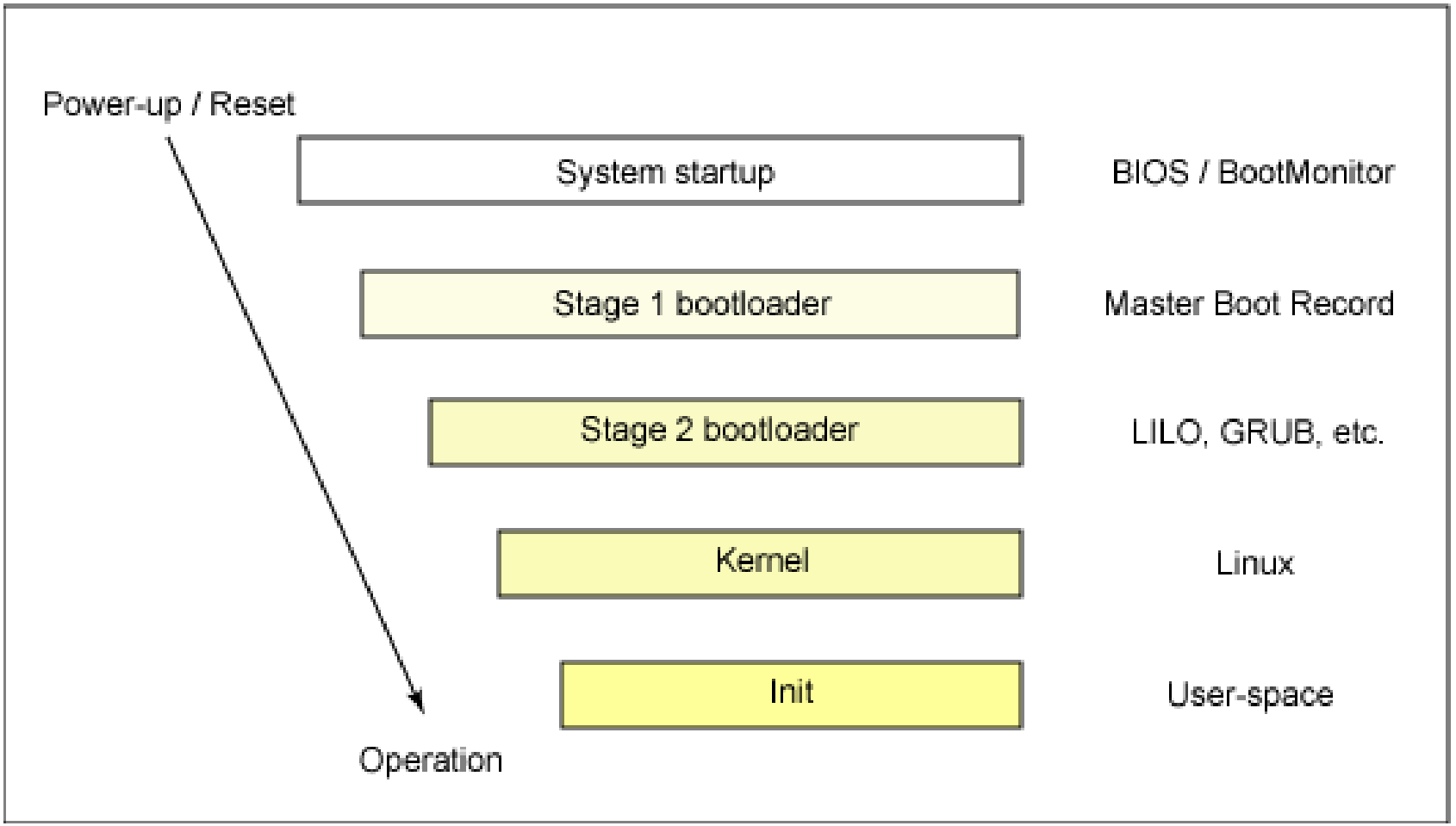
Processos no Linux

Hierarquia de Processos

- O Linux trabalha com **Hierarquia de Processos**.
  - Processos podem gerar outros processos, dando a nomenclatura de processo ”pai” e processo “filho”
- Processos filhos possuem apenas um processo pai,
- Um processo pai pode ter vários processos filhos.
- O fato de ser um processo filho não impede que o mesmo também tenha processos filhos.
- Este tipo de organização dificulta a propagação de vírus em nossos sistemas operacionais,
  - Quando um processo pai é “morto”, seja pelo sistema ou pelo próprio usuário, todos que estiverem abaixo dele na hierarquia serão mortos também.



Inicialização de Sistemas Linux



Processo *init*

- O processor ***init*** é o pai de todos os processos.
- Por padão, o processo *init* é identificado no sistema com o número 1, ou seja, o ID do processo *init* é 1
- Não pode ser abortado.
- Define em que níveis de execução (**runlevel**) o Linux inicializará a sequencia de scripts de inicialização:
  - Serviços do sistema.
  - Programa agetty ou mingetty
    - Configuram os terminais e viabilizam o login.
- Diretórios (Ubuntu):
  - /etc/init/
  - /etc/init.d/
  - /etc/rcX.d/

Processo *init*

- **Runlevel**:
  - Nível de execução do sistema Linux depois do boot.
  - O runlevel indica o modo de operação atual da máquina, definindo quais serviços e recursos devem permanecer ativos.
  - Tabelas por distribuição:
    - http://en.wikipedia.org/wiki/Runlevel
  - Comandos:
    - `runlevel`
    - `init [nível]`

Classificação dos Processos

- **Quanto a Execução:**
  - **Foreground** (primeiro plano):
    - São inicializados no terminal de comandos.
    - Exibem suas execução no monitor de vídeo.
    - Podem interagir com os usuários.
    - Prendem o prompt, **impedido** que outros processos sejam inicializados pelo mesmo terminal.
  - **Background** (segundo plano):
    - São inicializados no terminal de comandos.
    - **Não** exibem suas execução no monitor de vídeo (geralmente usa arquivos de log).
    - **Não** podem interagir com os usuários.
    - **Não** prendem o prompt, **permitindo** que outros processos sejam inicializados pelo mesmo terminal.

Classificação dos Processos

- Quanto ao Função
  - **Interativos**:
    - Iniciados a partir de uma sessão de usuário no terminal de comandos e controlados por ele.
    - Executa em foreground.
    - Recebe dados da entrada-padrão (**stdin**) do terminal, processa os dados e os envia para a saída-padrão (**stdout**) ou para a saída-padrão de erros (**stderr**).
  - **Batch (Em lote)**:
    - Processos em lote.
    - Controlados pelos comandos *at*, *batch* e *cron*.
  - **Daemons**:
    - Processos servidores.
    - Normalmente executados quando o Linux é inicializado.
    - Aguarda a requisição de serviços em background.

Atributos de um Processo

- **ID do processo (PID):** é um número inteiro que identifica cada processo em execução
  - Todo processo possui um PID que o identifica .
  - Único por processo.
- **ID do usuário (UID):** identificam o usuário que iniciou o processo;
- **ID do grupo (GID):** Esses atributos identificam grupo que o usuário pertence;
- **ID do processo pai (PPID):** Identifica o processo-pai que iniciou o processo em questão.
- **Tempo de vida** – define o tempo que este processo ficou em execução,
  - Existe processos com tempo de vida curto, outros processos com tempo de vida mais longo e existem processos que executam desde quando o computador é ligado até que seja desligado

Localização dos Processos

- No diretório **/proc** é criado um subdiretórios para cada processo em execução.
- Os nomes desses subdiretórios são os PIDs desses processos.
- Arquivos importantes:
  - cmdline;
  - environ;
  - status.

Visualizando os Processos

ps

- Visualizar quais processos estão sendo executados no computador. Além disso, mostra qual usuário executou o programa, hora que o processo foi iniciado, etc.

```
ps [opções]
```

Opções:

-a : Mostra os processos criados por você e de outros usuários do sistema.

-x : Mostra processos que não são controlados pelo terminal.

-u : Mostra o nome de usuário que iniciou o processo e hora em que o processo foi iniciado.

pstree

- Exibe informações sobre os processos ativos em forma de árvore.

```
pstree [opções] [pid|usuário]
```

pid – iniciar deste pido. Por padrão, é 1 (init)

usuário – mostra apenas árvores originadas de processos deste usuário.

Opções:

-a : Exibe argumentos de linha de comando.

-p : Mostra ids dos processos.

top

- Visualizar quais processos estão sendo executados no computador. Além disso, mostra qual usuário executou o programa, hora que o processo foi iniciado, etc.

```
top [opções]
```

Opções:

-d [tempo] : Atualiza a tela após o [tempo] (em segundos).

Abaixo algumas teclas úteis:

espaço : Atualiza imediatamente a tela.

Tecla Q: Sai do programa.

Executando Processos

Executando Processos

- **Pré-requisitos**: Permissão de execução (x);
- **Extensões**: **.sh** ou sem extensão.
- **2 formas** de executar um programa em **foreground**:
  - **. /** na frente do comando.
  - Apenas digitar o nome do comando.
    - O endereço do executável deve estar na variável de ambiente PATH.
    - Visualizar a variável PATH – ***echo $PATH***
    - ***Opa Professor! O que é uma variável de ambiente?***

Falando sobre Variáveis de Ambiente

Variáveis de Ambiente

- O **shell** é executado no sistema controlado por variáveis de ambiente.
- Variáveis de ambiente são definições e valores que o shell e outros programas utilizam para configuração no momento em que se realiza o login.
  - Estão localizada em arquivos de configuração que variam de shell para shell.
- 2 tipos:
  - **Variáveis de Ambiente Locais**: disponíveis somente pelo shell corrente, não sendo acessado pelos subprocessos.
    - Arquivo no Ubuntu: **/etc/profile**
  - **Variáveis de Ambiente Globais**: estão disponíveis tanto para o shell corrente como para os subprocessos.
    - Arquivo no Ubuntu: **~/.bash_profile**

Exemplos

- **PATH** - Esta é a variável de ambiente que define quais diretórios pesquisar e a ordem na qual eles são pesquisados para encontrar um determinado comando, para saber como o sistema faz esta pesquisa e quais diretórios ele procura um comando use o comando "echo $PATH".
- **HOME** - Esta variável identifica o diretório do usuário doméstico, use o comando echo $HOME para saber qual é o seu diretório HOME.
- **SHELL** - Esta variável identifica qual shell está sendo usado, use o comando echo $SHELL para saber qual é o shell que o seu sistema está usando.
- **TERM** - Esta variável define o tipo de terminal que está sendo usado, use o comando echo $TERM para saber qual o tipo de terminal está sendo usado pelo sistema.
- **USER** - Pré-define o nome de conta como variável de ambiente, ou seja, ao se logar ao sistema a ID do usuário é combinada com um nome de conta, para saber qual é o usuário corrente use o comando "echo USER".
- **LOGNAME** - Esta variável é um sinônimo para USER. Para saber qual é o seu logname use o comando "echo $LOGNAME".
- **OSTYPE** - Essa variável define o tipo de sistema operacional em uso. Para saber qual é o sistema operacional em uso use o comando "echo $OSTYPE".

Manipulando Variáveis de Ambiente

- Para visualizar as variáveis de ambiente no sistema pode-se utilizar os comandos a seguir.

Variáveis locais:

```
set
```

Variáveis globais:

```
env
```

ou

```
printenv
```

Manipulando Variáveis de Ambiente

- Para atribuir um valor a uma variável local veja o exemplo:

```
LINUX=free
```

- O comando *echo* exibe o valor de uma variável de ambiente

```
echo $LINUX
```

Manipulando Variáveis de Ambiente

- Agora vamos tornar esta variável local em uma variável global. Para isso devemos usar o comando export:

```
export LINUX
```

- Para deletar uma variável de ambiente da memória usamos o comando *unset*:

```
unset LINUX
```

Voltando...

Executando Processos

- Executando um processo em **Background**
  - Digitar o **&** no final do comando

- Executuntando uma sequêcia de comandos
  - Separar os comandos utilizando o **;**
  - Exemplo:
    - ***mkdir teste ; ls***

Controlando a Execução dos Processos

Processos em Primeiro Planos

- Abortando a execução de um processo:
  - Vai para o estado Encerrado.
  - **CTRL+C.**
- Suspendendo a execução de um processo:
  - Vai para o estado Bloqueado.
  - **CTRL+Z.**

kill

- Permite enviar um sinal de término a um comando/programa.

```
kill [número]
```

Onde:

número É o número de identificação do processo (PID), usando o **ps**.

Observações:

Você precisa ser o **dono do processo** ou o **usuário root** para terminá-lo ou destruí-lo.

Você pode verificar se o processo foi finalizado através do comando ps

killall

- Permite finalizar processos através do nome.

```
killall [processo]
```

Onde:

processo Nome do processo que deseja finalizar.

Observações:

- Você precisa ser o **dono do processo** ou o **usuário root** para terminá-lo ou destruí-lo.
- Você pode verificar se o processo foi finalizado através do comando ps.

jobs

- Mostra os processos que estão suspensos ou rodando em segundo plano (&).

```
jobs
```

fg

- Permite fazer um programa rodando em segundo plano ou parado, rodar em primeiro plano.

```
fg [número]
```

Onde:

número é o número obtido através do comando jobs.

Observação:

- Caso seja usado sem parâmetros, o fg utilizará o último programa interrompido (o maior número obtido com o comando jobs).

bg

- Permite fazer um programa rodando em primeiro plano ou parado, rodar em segundo plano.
  - `bg [número]`

Onde:

número número do programa obtido com o pressionamento das teclas CTRL+Z ou através do comando jobs.

Observação:

- Para fazer um programa em primeiro plano rodar em segundo, é necessário primeiro interromper a execução do comando com **CTRL+ Z**, será mostrado o número da tarefa interrompida, use este número com o comando bg para iniciar a execução do comando em segundo plano.

Definindo prioridades

- Um processo pode ter prioridade variando entre -20 (maior prioridade) e 19 (menor prioridade).
- A prioridade padrão é 10.
- Um usuário não-root pode apenas reduzir a prioridade de seus processos (aumentando o valor positivamente)
  - Não consegue retornar nem ao valor original.
- Comandos:
  - nice
  - renice

nice

- Executa um processo com um prioridade diferente (nice).

```
nice [opções] prioridade comando
```

Onde

prioridade: valor da prioridade (-19 a 20).

comando: comando a ser executado.

Opções:

--version exibe a versão do comando

renice

- Modifica a prioridade de um ou mais processos em execução.

```
renice prioridade [opções]
```

Onde

prioridade: valor da prioridade (-19 a 20).

Opções:

-p [pid]	altera a prioridade para um número de processo.
-u usuário	altera a prioridade para os processos de um determinado usuário.
-g grupo	altera a prioridade para os processos de um determinado grupo.