Capítulo 3: Gerenciamento de Processos e *Multiplexadores*

Type	Leitura
Materials	Capítulo 3.pdf
Reviewed	☐

GERENCIAMENTO DE PROCESSOS

- ***Programa***: sequencia de instruções que geralmente esta no arquivo e é armazenado.Código fonte e que não faz nenhuma execução
- ***Processo***: programa em execução. Diamons são programas que executão em segundo plano, assim que você liga o PC etc.
- **Runlevel**:
  - **Runlevel** refere-se aos diferentes modos de operação em que o sistema Linux pode estar, variando de 0 a 6. Cada nível define um estado do sistema, como desligamento, modo de usuário único, ou modo multiusuário com ou sem interface gráfica.
  - `Runlevel 0` geralmente é o estado de desligamento, e o `init 6` é usado para reiniciar o sistema.
  - Se você perder a senha do root, é possível iniciar o sistema em modo de usuário único (geralmente runlevel 1) para tentar recuperá-la.
- **Init**:
  - **Init** é o primeiro processo iniciado pelo kernel do Linux após o boot, sendo considerado o "pai de todos os processos". Ele é responsável por iniciar outros processos e serviços no sistema.
  - No Ubuntu, os principais diretórios relacionados ao `init` são:

- **/etc/init**: Contém scripts e configurações relacionados ao sistema de inicialização.
- **/etc/init.d**: Diretório contendo scripts de inicialização para serviços.
- **/etc/rcX.d/**: Contém links simbólicos para scripts em `/etc/init.d/`, onde `X` varia conforme o runlevel.

CLASSIFICAÇÃO DOS PROCESSOS

- Tem dois tipos de execução:
  1. ***Foreground (primeiro plano):***
     - interagem diretamente com o usuário e ocupam o terminal até que sua execução seja concluída.
  2. ***Background (segundo plano):***
     - Executados sem interação direta com o usuário e sem ocupar o terminal, permitindo que o usuário continue a realizar outras tarefas.Não exibem suas execuções (geralmente usa arquivos de logs)
     - ***Exemplos***: Executar um comando seguido de `&` (como `./script.sh &`) faz com que ele seja executado em background, liberando o terminal para outras tarefas enquanto o processo continua em execução.

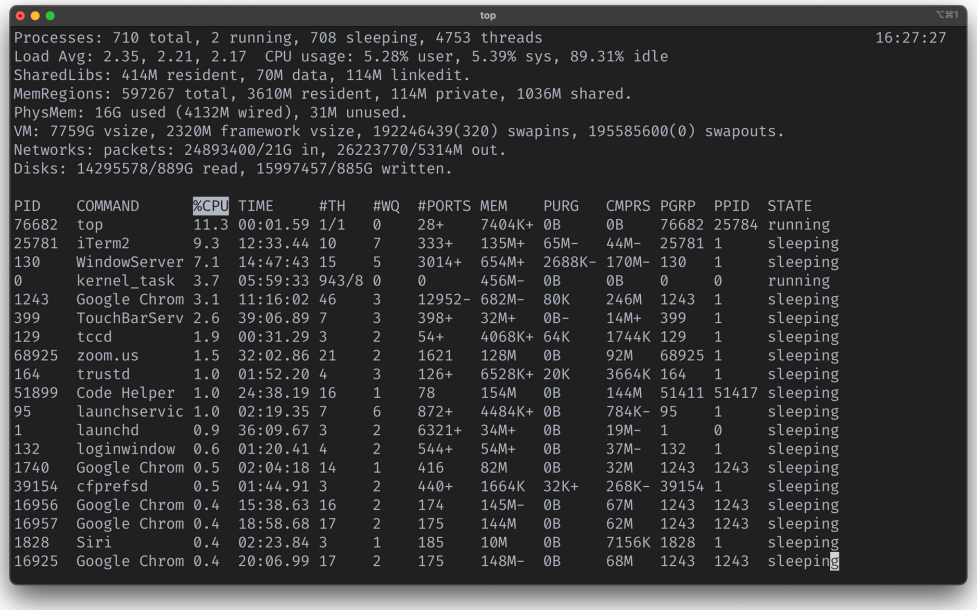
Controlando a Execução dos Processos

- Arquivos em segundo plano
  - `jobs`
- Vai para o estado encerrado
  - `CTRL + C`
- Vai para o estado bloqueado
  - `CTRL + Z`

- Para voltar, e quiser executar em primeiro plano, digite `fg` e em segundo plano `bg 1`

ATRIBUTOS DE UM PROCESSO

- No diretório /proc é onde tem todos os processos em execução
  - ***PID (Process ID):***
    - Identificador único do processo.
    - **Exemplo:** O processo `bash` pode ter o PID 1234.
  - ***UID (User ID):***
    - Identificador do usuário que iniciou o processo.
    - **Exemplo:** O UID 1000 pode representar o usuário "allyson".
  - ***GID (Group ID):***
    - Identificador do grupo ao qual o usuário pertence.
    - **Exemplo:** O GID 1000 pode representar o grupo "users".
  - ***PPID (Parent Process ID):***
    - Identificador do processo pai.
    - **Exemplo:** Se o `bash` iniciou o processo, o PPID pode ser 5678 (PID do `bash`).
  - ***Tempo de Vida:***
    - Tempo desde que o processo foi iniciado.
    - **Exemplo:** Um processo pode estar em execução há 5 minutos.



AUMENTO E DIMINUIÇÃO DE PRIORIDADE

- `nice -n 10` - aumentar a prioridade de um processo.A prioridade vai de -20 (maior) a 19 (menor)
- `renice -n 10 -p 597` - diminuir
- `renive -n 10 -u ou -g guiafoca` - diminuir a prioridade de um processo

PS

- `Tecla m` (para modificar a interface onde mostra a memória)
- `Apertar a` `tecla 1` para mostrar, em vez da soma das cpus, e sim de cada cpu.
- `SHIFT + M` (para classificar por uso de memória).
- `Tecla 'F', ele traz os nomes descritivos - o A e o W para ir e voltar, e Q para voltar ao normal -`.
- `Pressione a letra 'K` ' para desejar qual processo quer matar.
- `Pressione a letra D` atualizar o top a cada Xs
- `SHIFT + W` - Para salvar as config toda vez que entrar

OBS:

1. `pgrep` é usado para buscar os IDs de processos (PIDs) que correspondem a um determinado nome

2. `psgrep -x`: A opção `-x` faz com que `pgrep` busque exatamente o nome do processo que eu desejo buscar

Exemplo:

```
ps -o stat= -p 1234
```

3.

```
if [ -z "$pid" ]; then
echo "O processo não está em execução."
fi
```

COMANDOS PARA O “PS”

- `ps -e` ou `ps -A`:
  - **Descrição:** Lista todos os processos em execução no sistema.

- `ps -f`:
  - **Descrição:** Mostra uma lista detalhada (full-format listing) dos processos, incluindo campos como UID, PID, PPID, C (uso de CPU), STIME (hora de início), TTY, TIME e CMD (comando).
  - **Exemplo:** `ps -f` mostra uma lista detalhada dos processos.
- `ps -u [username]`:

    - **Descrição:** Exibe os processos pertencentes a um usuário específico.
    - **Exemplo:** `ps -u allyson` mostra todos os processos iniciados pelo usuário "allyson".
- `ps -aux`:
    - **Descrição:** Mostra todos os processos em execução, incluindo aqueles que não têm terminal (processos de fundo), e exibe informações detalhadas sobre cada um.
    - **Exemplo:** `ps -aux` é um comando comum para visualizar todos os processos com detalhes.
- `ps -l`:
    - **Descrição:** Exibe uma lista longa de processos, incluindo informações adicionais como o estado do processo (S, R, D, Z, T), prioridade e nice value.
    - **Exemplo:** `ps -l` mostra uma lista detalhada com mais atributos de cada processo.
- `ps -p [PID]`:
    - **Descrição:** Exibe informações sobre um processo específico, identificado pelo seu PID.
    - **Exemplo:** `ps -p 1234` mostra detalhes apenas do processo com PID 1234.
- `ps -T`:
    - **Descrição:** Exibe todos os processos relacionados ao terminal atual.
    - **Exemplo:** `ps -T` mostra os processos que estão associados ao terminal de onde o comando foi executado.
- `ps -C [nome_do_comando]`:
    - **Descrição:** Mostra os processos que correspondem a um comando específico.
    - **Exemplo:** `ps -C apache2` mostra todos os processos do servidor Apache.
- `ps aux|grep sleep` - procurar o sleep pra ve se ele tá com esse prioridade
- `tload` - monitorar os processos de maneira gráfica de uso de cpu etc.
- `vmstat 1` - monitorar linha a linha a cada 1s

COMANDOS PARA O “pstree”

`pstree [opções] [pid|usuário]`

Opções:

- `p`: Mostra os PIDs (Process IDs) ao lado dos nomes dos processos.
- `u`: Mostra o nome do usuário ao lado dos processos.
- `n`: Classifica os processos por PID em vez de ordem alfabética.
- `c`: Não agrupa processos idênticos sob um mesmo ramo.
- `a`: Mostra os argumentos completos dos processos.

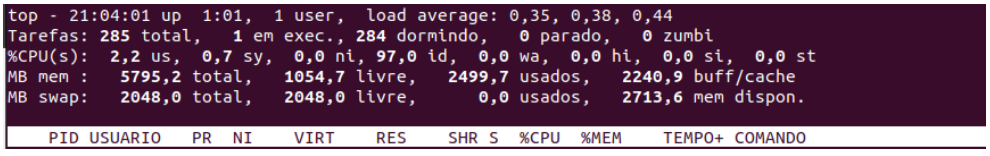
CASO EU QUEIRA EXECUTAR MAIS DE UM PROCESSO AO MESMO TEMPO

```
**`ls ; tree ; sleep 1`**

**`ls ; echo "Segundo comando" ; sleep 4 ; scho "Terceiro com

**`ps -aux`**

**`ps -ax`**

**`ps -axm|less`** #mostra o resultado de uso da memória de c

**`pf -aux -f`**  #processos mostrando em formato árvore

**`ps -axe`** - #mostra as variáveis de ambiente

**`ps -axew`** - #mostra as variaveis de ambiente e que as li

**`ps ax --sort=pid ou ps ax --sort=pid|head`** #classificar

**`pidof sshd`** - #lista o pid do processo em execução
```

```
**`pstree -c, -h, -p`** #lista os processos e o pid para cada

**`pstree -p -H 352`** - #destaca um processo específico em c

**`kill -9 1847`** - o '-9' #ele mata na hora, sem nem espera

**`kill -HUP numero_do_processo` - #recarregue seus arquivos

**`pgrep firefox`** - #encontrar os PIDs de processos

**`killall5 -9`** #vai matar todos os processos

**`nohup`** #continue funcionando mesmo se você fechar a jane
```

TOP

- `top -n` : pra voce especificar um tempo especifico
  - `Ex:  monitoramento=$(top -b -n "$tempo" | head -n 20)`
- `ps -d [tempo ]` : Atualiza a tela após o [tempo] (em segundos).



- ***us**: Tempo de CPU gasto executando programas de usuário.*
- ***sy**: Tempo de CPU gasto no kernel (sistema).*
- ***ni**: Tempo de CPU gasto em processos de baixa prioridade.*
- ***id**: Tempo de CPU ocioso (sem fazer nada).*
- ***wa**: Tempo de CPU esperando por I/O (disco).*
- ***hi**: Tempo de CPU lidando com interrupções de hardware.*
- ***si**: Tempo de CPU lidando com interrupções de software.*

- ***st**: Tempo de CPU "roubado" por uma máquina virtual enquanto o hypervisor serve outro processador.*

***OBS**: htop é pra ser instalado que é uma melhoria do top*

KILL

- Mata um programa em execução

```
kill [opções] PID
```

- **PID**: O identificador do processo que você deseja afetar.

Sinais Comuns

1. `SIGTERM` **(15)**: Este é o sinal padrão. Solicita que o processo termine de forma graciosa.

   ```
   kill PID
   ```

2. `SIGKILL` **(9)**: Força a terminação imediata do processo, sem a chance de ele limpar recursos ou salvar estado.

   ```
   kill -9 PID
   ```

3. `SIGINT` **(2)**: Interrompe um processo. Semelhante ao pressionar `Ctrl+C` no terminal.

   ```
   kill -2 PID
   ```

4. `SIGQUIT` **(3)**: Interrompe e gera um arquivo de despejo de núcleo (core dump) do processo.

```
kill -3 PID
```

MULTIPLEXADORES DE TERMINAIS

- Abrir várias sessões de terminal em uma única tela ou shell, permitindo alternar entre elas e realizar diferentes tarefas simultaneamente. Isso é especialmente útil quando se trabalha remotamente, gerenciando vários processos ou servidores em uma única interface de linha de comando.
- Dois dos multiplexadores de terminais mais populares no Linux são o GNU Screen e o tmux.

GNU Screen:

- Para iniciar o GNU Screen, basta digitar screen no terminal.
- Alguns comandos úteis dentro do GNU Screen incluem:
- Criar uma nova janela: Ctrl + A, C
- Altera de um para outra: Ctrl + A + A
- Altera para uma janela específica: Ctrl + A + 0, 1, 2..
- Alternar para a próxima janela: Ctrl + A, N
- Alternar para a janela anterior: Ctrl + A, P
- Lista as screens: screen ls
- Reconecta de volta para screen: screen -x
- Listar janelas: Ctrl + A, "

TMUX:

Ele mantem a barra de status e vê qual processo que tá rodando no terminal

- tmux ls - Listar as conexões criadas
- Criar uma nova janela: Ctrl + B, C
- Altera para uma janela específica: Ctrl + B + 0, 1, 2..
- Alternar para a próxima janela: Ctrl + B, N

- Alternar para a janela anterior: Ctrl + B, P
- Desconectar a screen: CTRL+B, D (desconectará você da sessão atual do tmux, mas a sessão continuará em execução em segundo plano.)
- Listar janelas: Ctrl + B, W

DÚVIDAS:

```
1. Diferença Entre $(...) e "${...}"

    $(...): É uma forma de substituição de comando. Você usa

data=$(date +%Y%m%d)

Isso executa o comando date +%Y%m%d e armazena o resultado na




   ${...}: É usado para referenciar variáveis dentro de uma s

backup="${diretorio}-${data}.tar.gz"

Aqui, ${diretorio} e ${data} são referências às variáveis dir




2. Verificar a existencia de um arquivo existente
```

```
if [ -f "$arquivo" ]; then



3. para identar debaixo do "if", precisa dar 4 espaços
```