

4. Gerenciamento de processos e tarefas

4.3 Ciclo de vida processos – gestão e hierarquia

4.4 Threads: conceitos, modelos e programação

Antes de iniciarmos vamos revisar as repostas da atividade aula anterior.

a) Na tela abaixo o que significa **Ss** e **TN** na coluna STAT, explique

```
root@DESKTOP-00I5LD3: ~
127 tty1 00:00:00 ps
root@DESKTOP-00I5LD3:~# ps -aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   8324    152 ?        Ss   14:38   0:00 /init
root         3  0.0  0.0   8328    152 tty1     Ss   14:38   0:00 /init
urubu100    4  0.0  0.0  15220   3740 tty1     S    14:38   0:00 -bash
urubu100   44  0.0  0.0  15160   3700 tty1     S    14:53   0:00 bash
urubu100   76  0.0  0.0  24688   6148 tty1     T    15:20   0:00 vi
urubu100   77  0.0  0.0  24452   5652 tty1     T    15:21   0:00 vi
root        81  0.0  0.0  14232   1844 tty1     S    15:22   0:00 su root
root        82  0.0  0.0  13808   2272 tty1     S    15:22   0:00 bash
root        92  0.0  0.0  24528   5760 tty1     T    15:22   0:00 vi
root        96  0.0  0.0  24512   5708 tty1     T    15:27   0:00 vi
root       117  0.0  0.0  24356   5656 tty1     TN   16:29   0:00 vi
root       119  0.0  0.0  15888   1932 tty1     T    16:30   0:00 top
root       120  0.0  0.0  24512   5744 tty1     TN   16:36   0:00 vi
root       121  0.1  0.0  15912   1976 tty1     T    16:36   0:01 top
root       128  0.0  0.0  15664   1860 tty1     R    17:00   0:00 ps -aux
root@DESKTOP-00I5LD3:~#
```

ESTADOS DOS PROCESSOS

D - Uninterruptible sleep (usually IO) – processo em modo sleeping ininterrupto (em geral relativos a E/S) – seminterrupção contínua.

R - Running or runnable (on run queue) – rodando ou em execução (na fila de execução).

S - Interruptible sleep (waiting for an event to complete) – Interrupção momentânea (em geral enquanto aguarda a conclusão de um evento.

T - Stopped, either by a job control signal or because it is being traced – interrompido por um sinal de controle ou por causa de algo que é rastreado.

X - dead (should never be seen) - morto

Z - Defunct ("zombie") process, terminated but not reaped by its parent. Processo morto, relativo ao processopai.

Os subcaracteres são:

N - low-priority (nice to other users) - baixa prioridade, fornecendo-a processos de outros usuários

L - has pages locked into memory (for real-time and custom IO) - mostra que há páginas bloqueadas na memória.

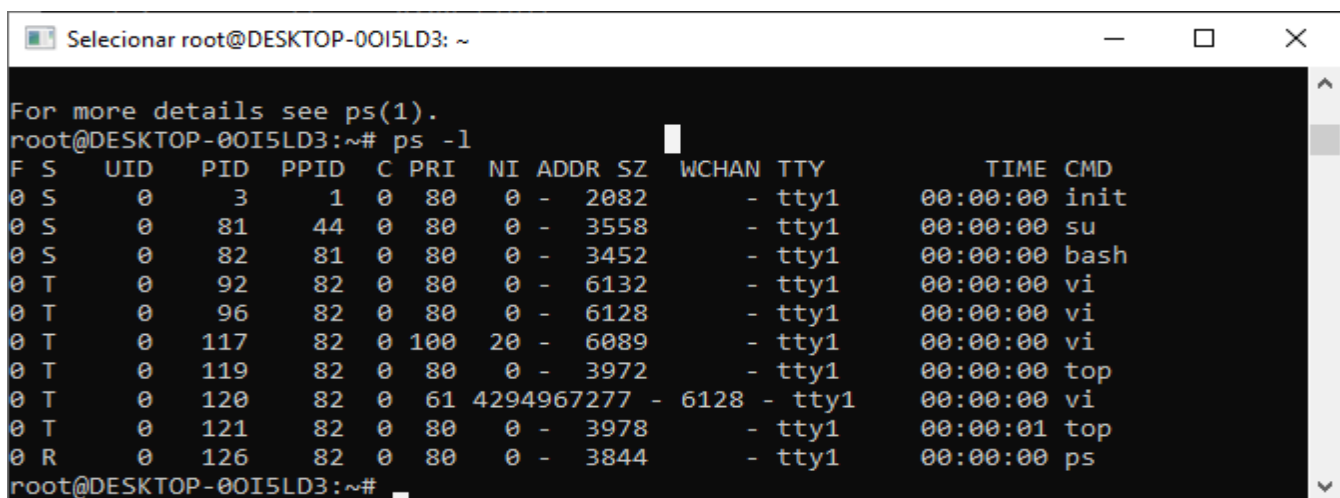
s - is a session leader - mostra se a sessão é líder. Exemplo do shell ou de um bash. Ambos são pai porque podem ser executados por diferentes grupos de usuários. Neste caso a sessão é o líder, se kill na sessão, kill em todos os processos filhos.

I - is multi-threaded - processos multithreadados

RSS é a sigla para *Resident Set Size*

VSZ é a *Virtual Memory Size*

b) O que significa ADDR SZ e WCHAN na tela abaixo, explique



```
For more details see ps(1).
root@DESKTOP-00I5LD3:~# ps -l
 F S      UID      PID      PPID      C  PRI      NI  ADDR  SZ  WCHAN  TTY      TIME  CMD
 0 S      0         3         1      0   80      0    -   2082    -   tty1    00:00:00 init
 0 S      0        81        44      0   80      0    -   3558    -   tty1    00:00:00 su
 0 S      0        82        81      0   80      0    -   3452    -   tty1    00:00:00 bash
 0 T      0        92        82      0   80      0    -   6132    -   tty1    00:00:00 vi
 0 T      0        96        82      0   80      0    -   6128    -   tty1    00:00:00 vi
 0 T      0       117        82      0  100     20    -   6089    -   tty1    00:00:00 vi
 0 T      0       119        82      0   80      0    -   3972    -   tty1    00:00:00 top
 0 T      0       120        82      0   61 4294967277 - 6128    -   tty1    00:00:00 vi
 0 T      0       121        82      0   80      0    -   3978    -   tty1    00:00:01 top
 0 R      0       126        82      0   80      0    -   3844    -   tty1    00:00:00 ps
root@DESKTOP-00I5LD3:~#
```

%CPU - Quanto da CPU o processo está usando

%MEM - Quanta memória o processo está usando

ADDR - Endereço de memória do processo

C ou CP - Informações de uso e agendamento da CPU

CMD - Nome do processo, incluindo argumentos, se houver

NI - nice

PID - Número de identificação do processo

PPID - Número de identificação do processo pai do processo

PRI - Prioridade do processo

TIME - tempo de uso total da CPU

TT ou TTY - Terminal associado ao processo

WCHAN (Waiting Channel) - (rotina do kernel para processo waiting relativo ao endereço de memória) Endereço de memória do evento pelo qual o processo está aguardando, processos em execução são marcados por um hífen (não há endereço fixo na memória para indicar)

Priority (PRI) x Nice (NI)

Um ponto que precisa ficar claro antes de seguirmos é a diferença entre PRIORIDADE (priority) e NICE. A **prioridade** de um processo é definida automaticamente e dinamicamente pelo kernel Linux.

O **NICE** é um atributo que permite ao administrador ou usuário influenciar a prioridade do processo. Quando usamos os comandos **nice** e **renice** para definir esse atributo, estamos definindo um NICE que irá consequentemente impactar a prioridade. Por padrão, o NICE de um processo é 0.

Voltando ao conteúdo:

4.3 Ciclo de vida processos – gestão e hierarquia

4.4 Threads: conceitos, modelos e programação

Composição de um processo

O sistema operacional lida com uma infinidade de processos, sendo necessário meios de controle.

Os processos apresentam um conjunto de características/atributos:

- Proprietário/grupo do processo;
- Estado do processo (em espera, em execução, etc);
- Prioridade de execução;
- Recursos de memória.

Grupo de processos

- Compartilhamento de recursos – Baseados em hierarquia de processos:
- Um processo pai cria processos filhos;
- Os filhos podem executar o mesmo código, ou trocá-lo;
- Obtem-se uma árvore de processos.
- Implica na definição da semântica de término de um processo:

Toda sua descendência morre

Os processos devem interagir com o disco para armazenar e recuperar dados não voláteis.

O disco físico é abstraído pelo Sistema de Arquivos, de acordo com uma hierarquia:

Diretórios e Arquivos.

Os diretórios estão frequentemente organizados de acordo com uma hierarquia em árvore:

- Raiz ('/')
- Diretório de trabalho de um processo ('.')
- Caminho relativo / absoluto
- Deve ter chamadas de sistema para acessar o Sistema de Arquivos.

Exemplos de chamadas de sistema aos processos:

Relativas ao Gerenciamento de processos: fork(), waitpid(), exit(), execve(...), getpid()...

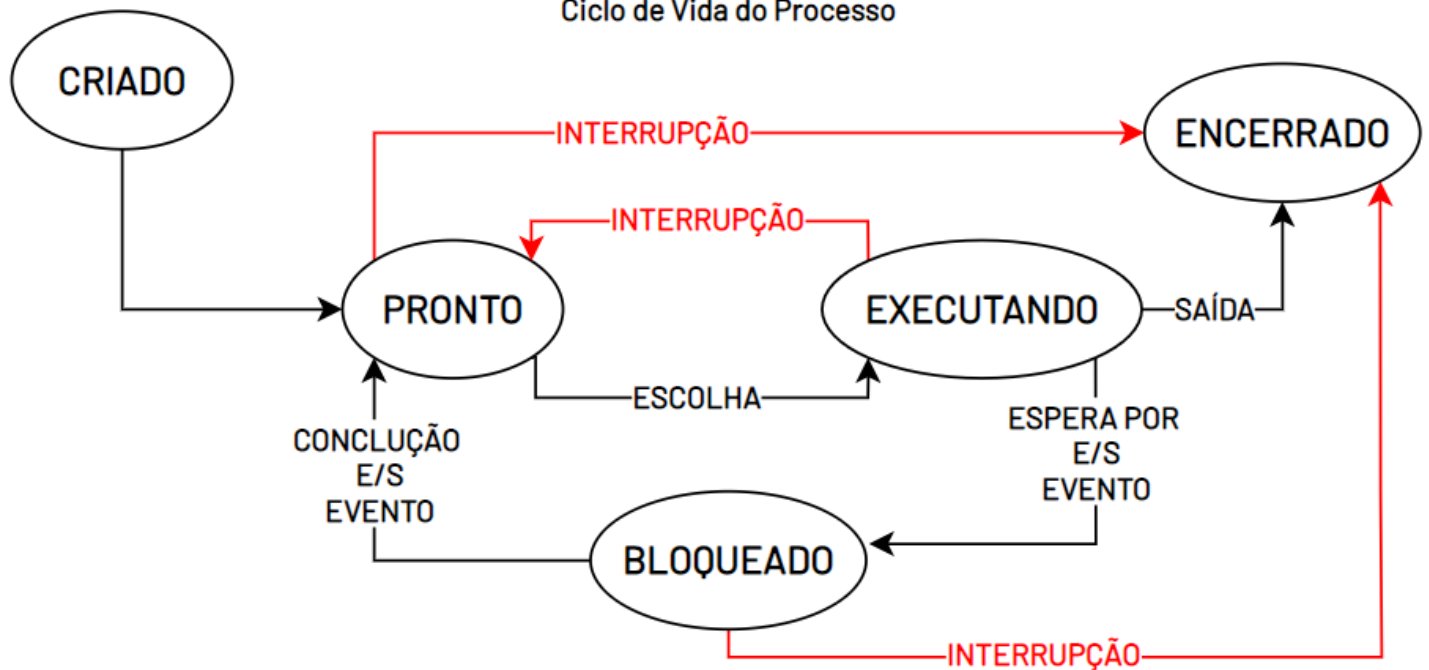
Relativas aos Sinais: sigaction(), sigreturn(), sigprocmask(), kill()...

Relativas ao Gerenciamento de arquivos: open(), close(), mknod(), read(), write(), pipe()...

Relativas aos Direitos de acesso: mkdir(), mount()...

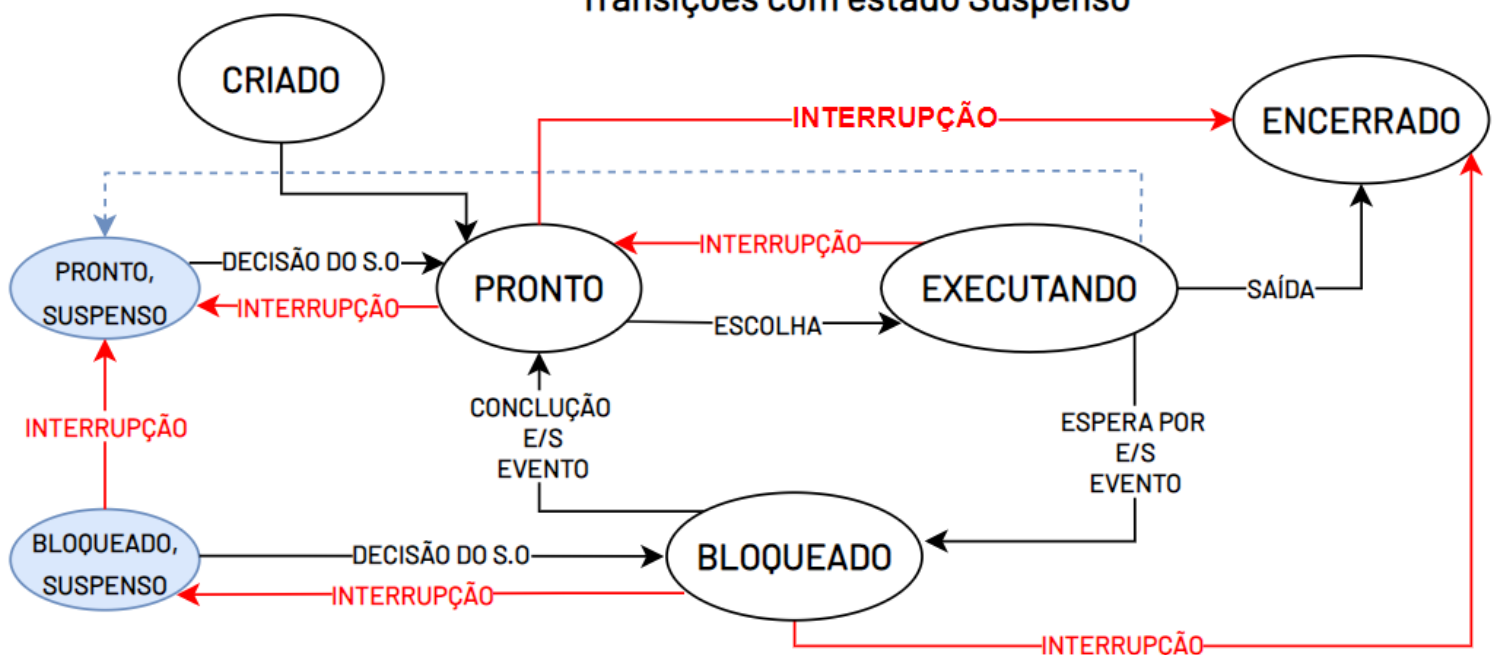
Relativas ao Gerenciamento de tempo: time()...

Diagrama de Estados
Ciclo de Vida do Processo



(Silberschatz, 4.1.2)

Transições com estado Suspenso



Agora vamos analisar as saídas do comando relativo as threads

ps -emo THREAD

Todas as saídas em hífen significam que estão em execução.

PRI – prioridade

SCNT – switch count

```
root@DESKTOP-00I5LD3: /lib# ps -emo THREAD
USER      %CPU PRI SCNT  WCHAN USER SYSTEM
root       0.0  -  -    -    -    -
root       0.0  19  -    -    -    -    -
root       0.0  -  -    -    -    -    -
root       0.0  19  -    -    -    -    -
urubu100   0.0  -  -    -    -    -    -
urubu100   0.0  19  -    -    -    -    -
urubu100   0.0  -  -    -    -    -    -
urubu100   0.0  19  -    -    -    -    -
urubu100   0.0  -  -    -    -    -    -
urubu100   0.0  19  -    -    -    -    -
urubu100   0.0  -  -    -    -    -    -
urubu100   0.0  19  -    -    -    -    -
root       0.0  -  -    -    -    -    -
root       0.0  19  -    -    -    -    -
root       0.0  -  -    -    -    -    -
root       0.0  19  -    -    -    -    -
root       0.0  -  -    -    -    -    -
root       0.0  19  -    -    -    -    -
root       0.0  -  -    -    -    -    -
root       0.0  19  -    -    -    -    -
root       0.0  -  -    -    -    -    -
root       0.0  -1  -    -    -    -    -
root       0.0  -  -    -    -    -    -
root       0.0  19  -    -    -    -    -
root       0.0  -  -    -    -    -    -
root       0.0  38  -    -    -    -    -
root       0.0  -  -    -    -    -    -
root       0.0  19  -    -    -    -    -
root       0.0  -  -    -    -    -    -
root       0.0  19  -    -    -    -    -
```

O kernel alterna entre threads em um esforço para compartilhar a *CPU* efetivamente; essa atividade é chamada de *alternância de contexto*.

Quando um thread é executado pela duração de seu intervalo de tempo ou quando é bloqueado porque requer um recurso indisponível no momento, o kernel encontra outro thread para executar e o contexto muda para ele.

O sistema também pode interromper o thread em execução no momento para executar um thread acionado por um evento assíncrono, como uma interrupção do dispositivo. Embora os dois cenários envolvam a alternância do contexto de execução da *CPU*, a alternância entre threads ocorre de forma *síncrona* em relação ao segmento atualmente em execução, enquanto as interrupções de manutenção ocorrem de maneira *assíncrona* em relação ao encadeamento atual.

Além disso, as alternâncias de contexto entre processos são classificadas como *voluntárias* ou *involuntárias*.

- Uma opção de contexto *voluntária* ocorre quando um encadeamento é bloqueado porque requer um recurso indisponível.
- Uma alternância *involuntária* de contexto ocorre quando um encadeamento é executado pelo período de tempo ou quando o sistema identifica um encadeamento de *prioridade mais alta* a ser executado.

Cada tipo de alternância de contexto é feito através de uma interface diferente.

A troca de contexto voluntária é iniciada com uma chamada para a rotina *sleep()*, enquanto uma troca de contexto involuntária é forçada pela invocação direta do mecanismo de troca de contexto de baixo nível incorporadas rotinas *mi_switch()* e *setrunnable()*.

A manipulação de eventos assíncronos é acionada pelo hardware subjacente e é efetivamente transparente para o sistema. Nossa discussão se concentrará em como o tratamento de eventos assíncronos se relaciona à sincronização do acesso às estruturas de dados do kernel.

pstree: esse comando mostra processos relacionados em formato de árvore. Sua sintaxe é:

pstree -opção PID

Entre as opções, tem-se:

- u - mostra o proprietário do processo;
- p - exibe o PID após o nome do processo;
- c - mostra a relação de processos ativos;
- G - usa determinados caracteres para exibir o resultado em um formato gráfico.

root@DESKTOP-00I5LD3: /home/marise

```
root@DESKTOP-00I5LD3:/home/marise# pstree
init—init—bash—su—bash—pstree
root@DESKTOP-00I5LD3:/home/marise# ps
  PID TTY          TIME CMD
    3 tty1      00:00:00 init
   35 tty1      00:00:00 su
   36 tty1      00:00:00 bash
  375 tty1      00:00:00 ps
root@DESKTOP-00I5LD3:/home/marise# pstree -u 3
init—bash(urubu100)—su(root)—bash—pstree
root@DESKTOP-00I5LD3:/home/marise#
```

root@DESKTOP-00I5LD3: /home/marise

```
init—bash(urubu100)—su(root)—bash—pstree
root@DESKTOP-00I5LD3:/home/marise# pstree -p 3
init(3)—bash(4)—su(35)—bash(36)—pstree(377)
root@DESKTOP-00I5LD3:/home/marise# pstree -p 35
su(35)—bash(36)—pstree(378)
root@DESKTOP-00I5LD3:/home/marise# pstree -p 36
bash(36)—pstree(379)
root@DESKTOP-00I5LD3:/home/marise# pstree -p 375
root@DESKTOP-00I5LD3:/home/marise#
```

root@DESKTOP-00I5LD3: /home/marise

```
root@DESKTOP-00I5LD3:/home/marise# pstree -c 3
init—bash—su—bash—pstree
root@DESKTOP-00I5LD3:/home/marise#
```

Selecionar root@DESKTOP-00I5LD3: ~

```
root@DESKTOP-00I5LD3:~# pstree -G 3
init—bash—su—bash—pstree
root@DESKTOP-00I5LD3:~#
```


Agora vamos visualizar as Threads:

Execute o comando top:

root@DESKTOP-00I5LD3: ~

```
top - 16:40:40 up 1:42, 0 users, load average: 0.52, 0.58, 0.59
Tasks: 6 total, 1 running, 5 sleeping, 0 stopped, 0 zombie
%Cpu(s): 15.6 us, 18.8 sy, 0.0 ni, 65.6 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 8321388 total, 1558416 free, 6526496 used, 236476 buff/cache
KiB Swap: 10215804 total, 9956196 free, 259608 used. 1654036 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	8324	152	128	S	0.0	0.0	0:00.15	init
3	root	20	0	8328	152	116	S	0.0	0.0	0:00.00	init
4	urubu100	20	0	15220	3728	3624	S	0.0	0.0	0:00.23	bash
35	root	20	0	14232	1836	1804	S	0.0	0.0	0:00.03	su
36	root	20	0	13828	2324	2236	S	0.0	0.0	0:00.38	bash
407	root	20	0	15900	1976	1412	R	0.0	0.0	0:00.11	top

Agora aperte shift

root@DESKTOP-00I5LD3: ~

```
top - 16:41:48 up 1:43, 0 users, load average: 0.52, 0.58, 0.59
Threads: 6 total, 1 running, 5 sleeping, 0 stopped, 0 zombie
%Cpu(s): 8.4 us, 9.5 sy, 0.0 ni, 81.9 id, 0.0 wa, 0.2 hi, 0.0 si, 0.0 st
KiB Mem : 8321388 total, 1584276 free, 6500636 used, 236476 buff/cache
KiB Swap: 10215804 total, 9961324 free, 254480 used. 1679896 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	8324	152	128	S	0.0	0.0	0:00.15	init
3	root	20	0	8328	152	116	S	0.0	0.0	0:00.00	init
4	urubu100	20	0	15220	3728	3624	S	0.0	0.0	0:00.23	bash
35	root	20	0	14232	1836	1804	S	0.0	0.0	0:00.03	su
36	root	20	0	13828	2324	2236	S	0.0	0.0	0:00.38	bash
407	root	20	0	15900	1976	1412	R	0.0	0.0	0:00.18	top