Capitulo 8.Monitoramento e gerenciamento de processos do Linux

Definição de um processo

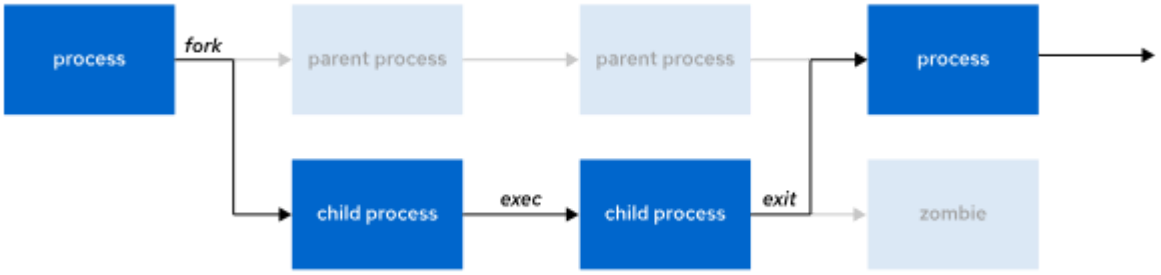
Um *processo* é uma instância em execução de um programa executável iniciado. A partir do momento em que um processo é criado, ele consiste nos seguintes itens:

- Um espaço de endereço de memória alocada
- Propriedades de segurança, incluindo privilégios e credenciais de proprietário
- Um ou mais threads de execução do código do programa
- Um estado de processo

O
ambiente de um processo é uma lista de informações que inclui os seguintes itens:

- Variáveis locais e globais
- Um contexto de agendamento atual
- Recursos alocados do sistema, como descritores de arquivo e portas de rede

Um processo *pai* existente duplica seu próprio espaço de endereços, que é conhecido como um processo *fork*, para criar uma estrutura de processo *filho*. Cada novo processo recebe uma única *ID de processo* (PID) para fins de rastreamento e segurança. A PID e a ID do processo pai (PPID) são elementos do novo ambiente de processo. Qualquer processo pode criar um processo *filho*. Todos os processos são descendentes do primeiro processo do sistema, `systemd`, em um sistema da Red Hat.



Por meio da rotina *fork*, um processo filho herda as identidades de segurança, os descritores de arquivo atuais e anteriores, os privilégios de porta e de recurso, as variáveis de ambiente e o código do programa. Um processo filho pode então executar seu próprio código do programa.

Normalmente, um processo pai *hiberna* quando o processo filho é executado, e configura uma solicitação de *espera* a ser sinalizada depois da conclusão do filho. Depois que o processo filho é encerrado, ele fecha ou descarta seus recursos e ambiente e deixa um recurso *zumbi*, que é uma entrada na tabela de processos. O pai, que é sinalizado como *ativo* quando o filho saiu, limpa a tabela de processo da entrada do filho, liberando, assim, o último recurso do processo do filho. O processo pai continua com sua própria execução de código do programa.

Descrição dos estados do processo

Em um sistema operacional multitarefa, cada CPU (ou núcleo da CPU) pode estar trabalhando em um processo por vez. À medida que um processo é executado, seus requisitos imediatos de tempo de CPU e alocação de recursos se alteram. Os processos recebem uma atribuição de *state*, a qual muda conforme a imposição das circunstâncias.

O diagrama e a tabela a seguir descrevem os estados do processo Linux em detalhes.

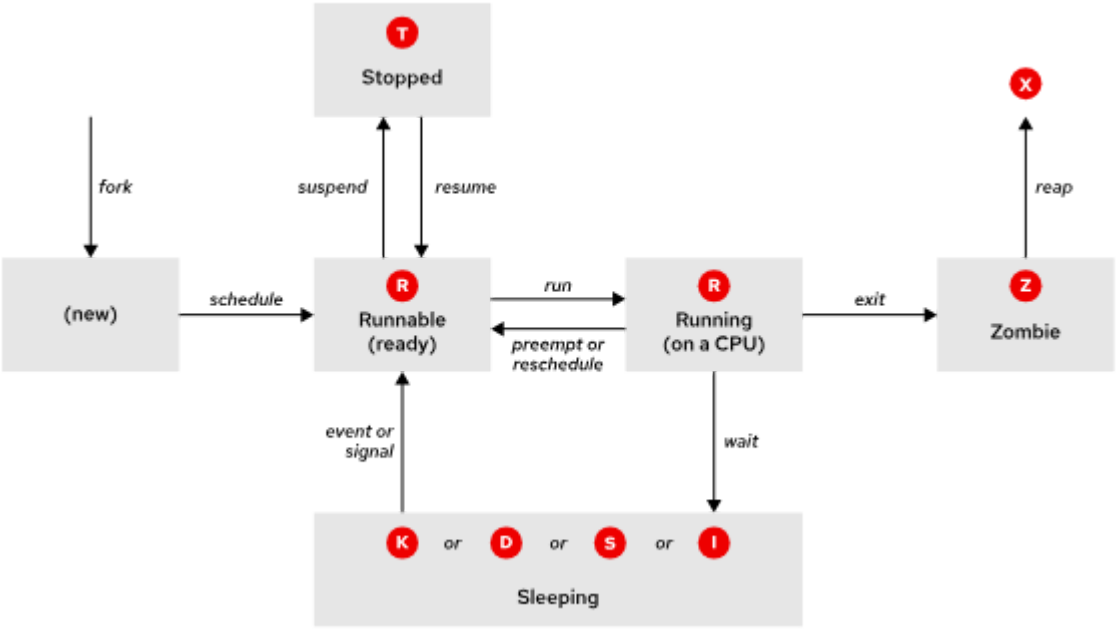


Tabela 8.1. Estados de processo no Linux

Name	Sinalizador	Descrição e nome de estado definidos no Kernel
Em execução	R	TASK_RUNNING: O processo está em execução em uma CPU ou aguardando para ser executado. O processo pode estar executando rotinas de usuário ou do kernel (chamadas do sistema) ou estar em fila e pronto quando no estado *Running* (ou *Runnable*).
Suspenso	S	TASK_INTERRUPTIBLE: o processo está esperando por alguma condição, como uma solicitação de hardware, um acesso a recursos do sistema ou um sinal. Quando um evento ou um sinal atende à condição, o processo retorna ao estado *Running*.
	D	TASK_UNINTERRUPTIBLE: este processo também está suspenso, mas, diferente daqueles em estado `S`, não responde aos sinais. É usado somente quando a interrupção do processo pode causar um estado de dispositivo imprevisível.
	K	TASK_KILLABLE: o mesmo que o estado ininterruptível `D`, mas modificado de maneira a permitir que uma tarefa em espera responda ao sinal para terminá-la (sair dela completamente). Utilitários frequentemente exibem processos *Termináveis* com um estado `D`.

	I	TASK_REPORT_IDLE: Um subconjunto do estado `D`. O kernel não conta esses processos ao calcular a média de carga. É usado para threads do kernel. Os sinalizadores TASK_UNINTERRUPTIBLE e TASK_NOLOAD são definidos. Semelhante a TASK_KILLABLE, também é um subconjunto do estado `D`. Aceita sinais fatais.
Interrompido	T	TASK_STOPPED: O processo foi interrompido (suspenso), geralmente por ser sinalizado por um usuário ou outro processo. O processo pode ser continuado por outro sinal para retornar à execução.
	T	TASK_TRACED: um processo que em depuração também está, temporariamente, interrompido e compartilha o mesmo sinalizador de estado `T`.
Zumbi	Z	EXIT_ZOMBIE: Um processo filho sinaliza para seu pai quando é concluído. Todos os recursos, exceto aqueles para a identidade do processo (PID), são liberados.
	X	EXIT_DEAD: Quando o pai limpa (extrai) a estrutura de processo remanescente do filho, o processo é liberado integralmente. Esse estado não pode ser observado em utilitários de listagem de processos.

Importância dos estados do processo

Ao solucionar problemas de um sistema, é importante entender como o kernel se comunica com os processos e como os processos se comunicam entre si. O sistema atribui um estado a cada novo processo. A coluna `S` do comando `top` ou a coluna `STAT` do comando `ps` exibem o estado de cada processo. Em um sistema de CPU único, apenas um processo pode ser executado por vez. É possível ver vários processos com um estado de `R`. No entanto, nem todos os processos são executados consecutivamente; alguns deles estão em estado de *espera*.

```
[user@host ~]$top
PID USER  PR  NI    VIRT    RES    SHR S  %CPU  %MEM   TIME
+    COMMAND
2259 root 20   0  270856  40316   8332 S   0.3   0.7   0:0
```

```
0.25  sssd_kcm
   1 root 20   0  171820  16140  10356 S   0.0   0.3   0:0
1.62  systemd
   2 root 20   0       0      0      0 S   0.0   0.0   0:0
0.00  kthreadd
...output omitted...
```

```
[user@host ~]$ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START
TIME COMMAND
...output omitted...
root         2  0.0  0.0      0     0 ?S    11:57   0:00 [k
threadd]
student   3448  0.0  0.2 266904  3836 pts/0R+   18:07   0:0
0 ps aux
...output omitted...
```

Use sinais para suspender, parar, retomar, encerrar ou interromper processos. Os processos podem capturar sinais do kernel, de outros processos e de outros usuários no mesmo sistema. Os sinais serão discutidos posteriormente neste capítulo.

Listagem de processos

O comando `ps` é usado para listar informações detalhadas para processos atuais.

- A identificação de usuário (UID) que determina os privilégios de processo
- A identificação de processo (PID) única
- Quantidade de CPU usada e tempo real decorrido
- Quantidade de memória alocada
- O local do processo `stdout`, conhecido como o terminal de controle
- O estado atual do processo

A opção `aux` do comando `ps` exibe todos os processos, incluindo processos sem um terminal de controle. Uma listagem longa (opções `lax`) fornece mais detalhes e oferece resultados mais rapidamente evitando pesquisas de nome de usuário. Uma sintaxe semelhante à do UNIX usa as opções `-ef` para exibir todos os processos. Nos exemplos a seguir, os threads de kernel agendados são exibidos entre colchetes no topo da lista.

```
[user@host ~]$ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START
TIME COMMAND
root           1  0.1  0.2 171820 16140 ?        Ss   16:47
0:01 /usr/lib/systemd/systemd ...
root           2  0.0  0.0      0     0 ?        S    16:47
0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   16:47
0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   16:47
0:00 [rcu_par_gp]
root           6  0.0  0.0      0     0 ?        I<   16:47
0:00 [kworker/0:0H-events_highpri]
...output omitted...
[user@host ~]$ps lax
F   UID     PID    PPID PRI  NI    VSZ   RSS WCHAN  STAT TT
Y  TIME COMMAND
4     0       1       0  20   0 171820 16140 -      Ss   ?
0:01 /usr/lib/systemd/systemd ...
1     0       2       0  20   0      0     0 -      S    ?
0:00 [kthreadd]
1     0       3       2   0 -20      0     0 -      I<   ?
0:00 [rcu_gp]
1     0       4       2   0 -20      0     0 -      I<   ?
0:00 [rcu_par_gp]
1     0       6       2   0 -20      0     0 -      I<   ?
0:00 [kworker/0:0H-events_highpri]
...output omitted...
[user@host ~]$ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1       0  0 16:47 ?        00:00:01 /usr/li
```

```
b/systemd/systemd ...
root           2       0  0 16:47 ?        00:00:00 [kthrea
dd]
root           3       2  0 16:47 ?        00:00:00 [rcu_g
p]
root           4       2  0 16:47 ?        00:00:00 [rcu_pa
r_gp]
root           6       2  0 16:47 ?        00:00:00 [kworke
r/0:0H-events_highpri]
...output omitted...
```

Por padrão, o comando `ps` sem opções seleciona todos os processos com a *ID de usuário efetiva* (EUID) do usuário atual e seleciona os processos associados ao terminal que está executando o comando. Os processos zumbis são listados com o rótulo `exiting` ou `defunct`.

Você pode usar a opção `--forest` do comando `ps` para exibir os processos em um formato de árvore para que você possa exibir as relações entre os processos pai e filho.

A saída padrão do comando `ps` é classificada por número de ID do processo. À primeira vista, a saída pode parecer usar ordem cronológica, mas o kernel reutiliza IDs de processo, portanto, a ordem é menos estruturada do que parece. Use as opções `-O` ou `--sort` do comando `ps` para classificar a saída. A ordem de exibição é equivalente à ordem da tabela de processos do sistema, a qual reutiliza as linhas na tabela à medida que os processos são extintos e gerados.

Controle de tarefas

Encerramento de processos

Exercício orientado: Encerramento de processos

Monitoramento de atividade de processo

Exercício orientado: Monitoramento de atividade de processo

Laboratório Aberto: Monitoramento e gerenciamento de processos do Linux