Eduardo Knabben Tiyo - 2551748
Pedro Chouery Grigolli - 2551845
Felipe Martins Sanches - 2390809
Ingrid Reupke Sbeguen Moran - 2349388

Manipulação de processos

Relatório técnico de atividade prática solicitado pelo professor Rodrigo Campiolo na disciplina de Sistemas Operacionais do Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Universidade Tecnológica Federal do Paraná – UTFPR
Departamento Acadêmico de Computação – DACOM
Bacharelado em Ciência da Computação – BCC

Campo Mourão
Abril / 2025

Resumo

Este relatório apresenta a análise do gerenciamento de processos no sistema operacional GNU/Linux, utilizando o comando `ps aux` para identificação de processos do sistema (daemons) e processos interativos. Foram analisadas informações como uso de CPU, memória e tempo de execução dos processos, além da verificação da existência de processos zumbis. Também foram explorados os comandos para suspensão e retomada de processos, e implementado um programa em `C` capaz de criar processos recursivamente, demonstrando os limites do sistema e os riscos de sobrecarga. Os resultados obtidos evidenciam a importância do monitoramento e controle de processos para garantir o desempenho e a estabilidade do ambiente no sistema operacional.

Palavras-chave: Processo. Daemon. Linux. Sistemas Operacionais. Manipulação.

Sumário

1 Introdução

Este relatório refere-se à compreensão e manipulação de processos do sistema operacional GNU/Linux. O relatório foi seccionado em: objetivos, fundamentação, materiais, procedimentos e resultados, discussão dos resultados e, por fim, conclusões.

2 Objetivos

- Compreender a estrutura de processos em Linux.
- Manipular processos em linguagem de programação.

3 Fundamentação

No contexto dos sistemas operacionais, um processo pode ser definido como um programa em execução. Ele representa a menor unidade de execução que o sistema gerencia de forma independente, com seu próprio espaço de memória, contador de programa, registradores e outros recursos. No sistema operacional GNU/Linux, a manipulação e o gerenciamento de processos são essenciais para o funcionamento multitarefa, permitindo que diversos programas sejam executados simultaneamente.

Daemons são processos que executam continuamente, sem interação com o usuário, para prover serviços ao sistema operacional, como gestão de impressoras, de conexões de rede, etc. A palavra daemon vem do grego antigo e significa “ser sobrenatural” ou “espírito”. (MAZIERO, 2023)

4 Materiais

- VirtualBox.
- Distribução GNU/Linux Debian 12.10.
- Kernel Linux 6.14.
- Intel Core i5-1135G7 @4.20GHz.
- Intel TigerLake-LP GT2 [Iris Xe Graphics].
- 8GB RAM DDR4 3200MHz.

5 Procedimentos e Resultados

5.1 Execute o comando ps aux e identifique três processos do sistema (daemons) e três processos interativos.

Processo do sistema/daemon

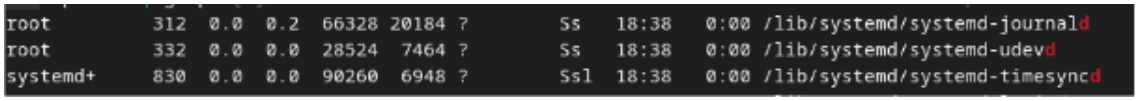

Figura 1 – ps aux - daemons

Processos Interativos

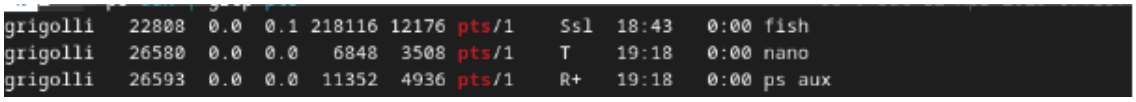

Figura 2 – ps aux - interativos

A sequência de saída do *ps aux* é:

- USER: usuário que iniciou o processo
- PID: Process ID do processo
- %CPU: Porcentagem de CPU sendo utilizada pelo processo
- %MEM: Porcentagem de RAM utilizada pelo processo
- VSZ: Tamanho virtual da memória usada (em KB)
- RSS: Resident Set Size - quanto de memória real (RAM) está sendo usada (em KB)
- TTY: Qual terminal está associado ao processo. Se for "?", significa que não está atrelada a nenhum terminal, portanto, são geralmente DAEMONS.
- STAT: Estado do processo
- START: Horário em que o processo foi iniciado
- TIME: Tempo total de CPU usado desde o início do processo
- COMMAND: O comando que iniciou o processo.

Ou seja, no processo DAEMON:

- USER: root
- PID: 312
- %CPU: 0.0
- %MEM: 0.2
- VSZ: 66328
- RSS: 20184
- TTY: ? (daemon)
- STAT: Ss
- START: 18:38
- TIME: 0:00
- COMMAND: /lib/systemd/systemd-journald

No processo INTERATIVO:

- USER: grigolli (usuário pessoal)
- PID: 22808
- %CPU: 0.0
- %MEM: 0.1
- VSZ: 218224
- RSS: 14216
- TTY: pts/1
- STAT: Ssl
- START: 18:43
- TIME:0:00
- COMMAND: fish

5.2 Há processos zombies executando em seu sistema operacional? Posso eliminá-los do sistema usando o comando *kill -SIGKILL pid_zombie? Justifique.*

Não existem processos zombies rodando naturalmente no sistema, visto que todos os processos, ao serem finalizados, viram zumbis por apenas alguns milésimos de segundos antes de serem "limpos" (*wait()* vindo do processo pai). Caso existam zombies no sistema de forma natural, sem ser a fim de testes, isso pode indicar que o sistema operacional apresente algum problema, e acabar os PIDs disponíveis para novos processos no futuro.

Não é possível encerrar o processo executando *kill -SIGKILL pid_zombie*, uma vez que esse comando faz com que o próprio sistema encerre o processo, porém o processo já está encerrado, a única coisa que não está encerrada é o espaço alocado para o processo (vulgo BCP) na Tabela de Controle de Processos.

5.3 Quais os processos com maior utilização de CPU? Quais os processos com maior utilização de memória? Qual o processo do usuário está a mais tempo em execução?

5.3.1 Processo com maior utilização de CPU

```
~ ps -eo pid,cmd,%mem,%cpu --sort=-%cpu | head
  PID CMD                         %MEM %CPU
 4136 ./jetbrains-toolbox --minim  3.5 17.1
 1806 /usr/libexec/packagekitd     0.5 12.4
 5157 /usr/lib/firefox-esr/firefo  5.1  7.4
 3869 /usr/bin/gnome-shell         2.7  4.8
 5659 /usr/libexec/gnome-terminal  0.6  1.8
 4146 /usr/bin/gnome-software --g  1.2  1.5
 5417 /usr/lib/firefox-esr/firefo  1.8  1.4
 4201 /usr/libexec/ibus-extension  0.3  1.4
 4181 /home/grigolli/.local/share  0.0  1.1
```

Figura 3 – Processo com maior utilização de CPU

5.3.2 Processo com maior utilização de memória

```
~ ps -eo pid,cmd,%mem,%cpu --sort=-%mem | head
  PID CMD                         %MEM %CPU
 5420 /usr/lib/firefox-esr/firefo  6.7 11.3
 5157 /usr/lib/firefox-esr/firefo  6.1 12.6
 4136 ./jetbrains-toolbox --minim  3.6  8.5
 3869 /usr/bin/gnome-shell         2.7  5.4
 5417 /usr/lib/firefox-esr/firefo  2.0  2.5
 5272 /usr/lib/firefox-esr/firefo  1.4  0.2
 4146 /usr/bin/gnome-software --g  1.2  0.7
 5325 /usr/lib/firefox-esr/firefo  1.2  0.2
 4127 /usr/libexec/evolution-data  1.0  0.1
```

Figura 4 – Processo com maior utilização de memória

5.3.3 Processo com maior tempo de execução

```
~ ps -eo pid,user,etime,cmd --sort=-lstart | head -n 10
  PID USER         ELAPSED CMD
 6397 grigolli       00:00 ps -eo pid,user,etime,cmd --sort=-lstart
 6398 grigolli       00:00 head -n 10
 6350 root           00:14 [kworker/6:0-events]
 6330 root           00:31 [kworker/7:0]
 6280 root           01:23 [kworker/2:1-mm_percpu_wq]
 6241 root           03:16 [kworker/u16:0-events_unbound]
 6137 root           05:01 [kworker/1:1]
 6136 root           05:10 [kworker/u17:0]
 6119 root           06:05 [kworker/4:0-mm_percpu_wq]
```

Figura 5 – Processo com mais tempo em execução

5.4 Como eu faço para suspender um processo no Linux? Como eu faço para retomar a execução novamente?

Existem várias maneiras de suspender um processo no Linux. A mais usada é por meio do comando *'kill -SIGTSTP $PID"* e para retornar a execução é por meio do comando *"kill -SIGCONT $PID"*

5.5 O que aconteceria se um processo criasse recursivamente processos filhos indefinidamente? Implemente um programa em Linux que faça isso e apresente o resultado.

Caso um processo crie recursivamente processos filhos indefinidamente, chegaria um momento que faltaria PIDs no Bloco de Controle de Tarefas, TCB (do inglês *Task*

Control Block), então geraria um erro *"fork(): resource temporarily unavailable"*. Provavelmente atingiria o número total de processos permitidos no sistema acarretando em travamentos parciais ou totais e até mesmo a morte de processos importantes caso o *OOM-killer* for ativado.
Programa em C:

Listing 1 – Exemplo de criação recursiva de processos

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
    while (1) {
        pid_t pid = fork();
        if (pid < 0) {
            perror("Erro ao criar processo");
            exit(1);
        }
    }
    return 0;
}
```

6 Discussão dos Resultados

A partir da execução do comando ps aux, foi possível observar na prática o funcionamento e a categorização dos processos no Linux. Os processos do sistema (daemons), identificados pelo usuário *root* e pela ausência de terminal associado (TTY *?*), evidenciam que são executados em segundo plano, responsáveis por manter serviços essenciais do sistema

Por outro lado, os processos interativos, vinculados a terminais específicos (como *pts/1*), pertencem a usuários comuns, como *grigolli*, e representam aplicações ou *shells* utilizados ativamente. No caso analisado, o processo *fish* (Friendly Interactive SHell) demonstrou características compatíveis com processos interativos.

Sobre os processos zumbis, não foram identificados no sistema, o que confirma o comportamento esperado: normalmente, processos zumbis existem apenas por milissegundos até que o processo pai execute *wait()* e libere a entrada da Tabela de Controle de Processos. Também ficou evidente que o comando *kill -SIGKILL* não pode eliminar um zumbi, pois o processo já não está mais ativo — apenas seu registro permanece.

A identificação dos processos com maior consumo de CPU e memória permitiu

verificar como o sistema prioriza e distribui recursos entre processos. Esse diagnóstico é importante para detectar pontos de sobrecarga e possíveis abusos de recursos. Da mesma forma, identificar o processo de maior tempo de execução ajuda a entender quais aplicações ou serviços permanecem ativos por mais tempo, o que pode indicar serviços críticos ou processos esquecidos.

Por fim, a implementação do programa em C que cria processos recursivamente demonstrou um cenário extremo de esgotamento de recursos, levando ao erro *fork(): resource temporarily unavailable*. Esse comportamento evidencia a importância dos limites impostos pelo sistema quanto à quantidade de processos ativos, como forma de manter a estabilidade e segurança do ambiente operacional.

7 Conclusões

Ao colocar em prática e examinar os comandos e testes sugeridos, foi possível captar, na prática, como a administração de processos funciona no GNU/Linux. A diferença entre os processos do sistema (daemons) e aqueles com os quais se interage ficou clara ao observar as características mostradas pelo comando ps aux, como o usuário ligado, o terminal (TTY) e o comando que deu início ao processo.

Também foi possível reforçar conceitos importantes sobre processos zumbis, entendendo seu comportamento transitório e a impossibilidade de eliminá-los diretamente, uma vez que já se encontram encerrados, restando apenas seus registros na tabela de processos.

A descoberta dos processos que mais consomem CPU, memória e tempo de uso trouxe à tona informações para ficar de olho no sistema, permitindo identificar pontos de sobrecarga e examinar como os recursos são divididos entre os processos ativos.

Além disso, a criação de um programa que gera processos em cadeia mostrou os perigos de se chegar ao limite de processos do sistema, o que pode prejudicar sua firmeza e acessibilidade.

Por fim, esse estudo deixou ainda mais clara a importância de entender a gestão de processos para diagnosticar problemas, manter o bom desempenho e garantir a segurança de ambientes operacionais que usam Linux.

8 Referências

MAZIERO, C. *Sistemas Operacionais: Conceitos e Mecanismos*. 2023. https://wiki.inf.ufpr.br/maziero/lib/exe/fetch.php?media=socm:socm-livro.pdf. Acesso em: abr. 2025. p. 208. Citado na página 4.