Matheus Vinicius da Costa - 28/03/2024

---

1 - Execute o comando ps aux e identifique três programas do sistema (daemons) e três programas do usuário, explicando os valores cada uma das colunas para um de cada tipo (sistema e usuário).

R:
> Daemons :  rtkit-daemon, chroot helper, accounts-daemon,.

> Sys/User : gnome-shell, firefox-esr,  pipewire.

Colunas:
• USER: Indica qual usuário iniciou o processo.
• PID: Identificador de processo.
• CPU%: Tempo de CPU.
• MEM%: Memória sendo usada pelo processador.
• RES: A memória não swap que uma tarefa usou.
• PRI: Prioridade do processo
• NI: Prioridade, nível amigável.
• VIRT: Memória Virtual.
• SHR: Memória compartilhada. Shared
• S ou STAT: Status do processo.
• Time: Tempo do processo desde sua init.
• Command: Nome do processo/comando.

---

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

R:
Não há processos zombies no momento, portanto o comando "kill -SIGKILL pid_zombie" não pode eliminar tais processos porque eles não existem ou já estão mortos.
O comando proprio é enviar o sinal SIGCHLD para o processo pai com > kill -s SIGCHLD [ppid]

ppid = pid do processo pai

---

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?

R:
• CPU: Firefox
• Memória: Firefox
• Tempo: Firefox

> Colocar filtro em cpu,mem e time
ps -eo pid,cmd,%mem,%cpu,time –sort=-%cpu
ps -eo pid,cmd,%mem,%cpu,time –sort=-%mem
ps -eo pid,cmd,%mem,%cpu,time –sort=-time

---

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

R:
• Suspender: kill -STOP [pid]
• Continuar: kill -CONT [pid]

---

5 - O que aconteceria se um processo criasse recursivamente processos filhos indefinidamente? Implemente um programa em Linux que faça isso e apresente o resultado. (Sugestão: testar na máquina virtual).

R:
Criar um programa que gera processos filhos indefinidamente resultaria em uma rápida exaustão dos recursos do sistema, já que cada processo filho continuaria a gerar mais filhos independentemente. Isso consumiria gradualmente toda a memória disponível e os recursos do sistema, levando eventualmente à terminação do programa devido à falta de recursos.

Code:

```
void infinite_fork() {
    pid_t pid = fork();
    if (pid > 0) {
        printf("Processo pai: PID %d\n", pid);
        wait(NULL);
    } else if (pid == 0) {
            printf("Processo filho: PID %d\n", pid);
            infinite_fork();
```

```
        } else {
            perror("Erro ao criar processo filho");
            exit(1);
        }
}
```