Universidade do Estado de Santa Catarina – Bacharelado em Ciência da Computação
TE em Ciência da Computação – Prof. Rafael R. Obelheiro

Exercícios – Objetivos 103.5 e 103.6

Parte 1

1. Abra uma janela de terminal, e use o comando `tty` para verificar o arquivo de dispositivo associado ao seu terminal.

2. Execute o comando `yes Teste1`; este comando exibe repetidamente os seus argumentos de linha de comando.

3. Pressione Ctrl-C para encerrar o processo.

4. Execute novamente o comando, mas agora em *background* e com a saída redirecionada para `/dev/null`:

   ```
   $ yes Teste1 >/dev/null &
   ```

5. Use o comando `fg` para trazer o processo para o primeiro plano (*foreground*).

6. Pressione Ctrl-Z para suspender o processo.

7. Coloque mais algumas instâncias em execução:

   ```
   $ yes Teste2 >/dev/null &
   $ yes Teste3 >/dev/null &
   ```

8. Agora existem três instâncias de `yes`, uma suspensa e duas em execução. Use o comando `jobs` para listar os processos que foram criados.

9. Abra duas abas na aplicação de terminal, e em cada uma execute o comando `tty`.

10. Em cada uma das abas novas, execute uma nova instância:

    ```
    $ yes Teste4 >/dev/null  # 2a aba
    $ yes Teste5 >/dev/null  # 3a aba
    ```

11. Volte para a primeira aba e use o comando `ps au` para listar os processos vinculados a terminais. Qual o estado de execução de cada instância de `yes`?

12. Ainda na primeira aba, use o comando `kill` *`PID`* para encerrar a instância de `yes Teste4` em execução na segunda aba (veja o PID desse processo na saída de `ps`).

13. Ainda na primeira aba, use o comando `kill -STOP` *`PID`* para parar (pausar) a instância de `yes Teste5` em execução na terceira aba (veja o PID desse processo na saída de `ps`). Volte para a terceira aba e confirme que o processo está parado.

14. Na primeira aba, use o comando `ps au` para verificar o estado de execução do processo suspenso.

15. Ainda na primeira aba, use o comando `kill -CONT` *`PID`* para que a instância de `yes Teste5` retome a execução na terceira aba, e depois o comando `ps au` para verificar se o estado de execução do processo foi alterado. Volte para a terceira aba para confirmar que a execução do processo foi retomada.

16. Na primeira aba, use o comando `killall yes` para encerrar todas as instâncias de `yes` ainda em execução, e depois o comando `ps au` para verificar se não restou nenhuma instância no sistema.

17. Na segunda aba, crie um arquivo `ping.out` usando o comando `touch`, e na sequência monitore esse arquivo com `tail -f ping.out`.

18. Na terceira aba, execute o comando abaixo:

```
$ ping 8.8.8.8 >ping.out &
```

19. Na primeira aba, use `pstree` ou `ps axjf` para observar a hierarquia de processos e descobrir quem é o processo pai de `ping`.

20. Observe o que acontece na segunda aba durante alguns segundos, depois feche a terceira aba. Ainda são mostradas novas linhas do arquivo na segunda aba?

21. Na primeira aba, use `ps` ou `top` para verificar se `ping` continua em execução. Caso `ping` continue ativo, encerre a sessão (i.e., faça logout/login) e verifique novamente.

22. Se você precisou reiniciar a sessão no passo anterior, repita o passo 17.

23. Abra novamente uma terceira aba, e nela execute o comando:

```
$ nohup ping 8.8.8.8 >ping.out &
```

24. Repita os passos 19 a 21. Houve alguma diferença?

25. Use o comando `top` para localizar o `ping`, e encerre sua execução.

Parte 2

1. O programa `prog`, disponível nos arquivos do Moodle, gera 3 processos. Um desses processos consome CPU, outro consome memória, e outro não consome recursos. Execute o programa e, na sequência, use `top` para identificar e encerrar os processos que consomem recursos em excesso.

2. Compile os programas `conta.c` e `conta-inf.c`, disponíveis nos arquivos do Moodle. Na sequência, execute os seguintes comandos:[1]

```
$ for i in {1..8} ; do ./conta-inf >/dev/null & done
$ time ./conta
```

   Quanto tempo o programa `conta` leva para executar?

3. Execute o comando abaixo:

```
$ time nice ./conta
```

   Há alguma diferença no comportamento de conta?

4. Enquanto `conta` executa, encerre 3 instâncias quaisquer de `conta-inf`. Qual o efeito disso sobre o comportamento de `conta`?

5. Após o término de `conta`, execute-o novamente (com `nice`). Quanto tempo leva a execução? Há alguma diferença em relação à primeira execução (sem `nice`)?

[1] Se você estiver usando uma máquina com um número de núcleos diferente de 8, ajuste o limite superior em `1..8`.

Parte 3

1. Em uma aba de terminal, execute o comando `top`, e observe durante alguns segundos as médias de carga (*load averages*) de 1, 5 e 15 minutos reportadas pela ferramenta.

2. Em uma segunda aba, inicie três instâncias de `conta-inf` executando três vezes o comando

   ```
   $ ./conta-inf >/dev/null &
   ```

   Retorne à primeira aba e observe o efeito da execução dessas instâncias sobre a carga. A média de carga de 1 minuto é maior ou menor que as médias de 5 e 15 minutos? Isso significa que a carga está aumentando ou diminuindo?

3. Na segunda aba, inicie novas instâncias de `conta-inf`, dando um intervalo de alguns segundos entre cada comando. Observe o efeito sobre a carga reportada por `top`.

4. Depois que o sistema estiver com mais de 10 instâncias de `conta-inf`, comece a encerrá-las, e observe como a carga do sistema é afetada. Encerre 2 ou 3 instâncias de cada vez; você pode encerrar uma instância (a mais recente) com o comando

   ```
   $ pkill -ne conta-inf
   ```