Roteiro de Atividades

1. Faça um resumo de todas as seções do Capítulo 3, do livro Advanced Linux Programming, e implemente os exemplos disponibilizados.

O capítulo 3 do livro **Advanced Linux Programming aborda um panôrama geral sobre os processos em Linux** e podemos resumir o capítulo abordando sobre cada um deles:

1. TOP

O comando *top* é a maneira mais comum de verificar o uso de processos do sistema e constatar quais deles estão consumindo mais memória ou processamento. Note que os primeiros itens da lista são os que mais consomem recursos do computador. Para cancelar a execução e voltar à linha de comando, basta pressionar a tecla *Q* ou a combinação *Ctrl+C*. É importante nessa relação que a primeira coluna, a coluna **PID**, exibe o número de identificação de determinado processo. É por meio desse número que você poderá, por exemplo, encerrar um processo com o comando *kill*.

2. PS

O comando *ps* lista os processos em execução no sistema. Porém, diferentemente do top, ele não traz informações sobre o quanto de processamento ou de memória ele está consumindo. Apesar disso, o ps é uma maneira bem mais ágil de consultar o PID de um processo, principalmente ao ser usado em conjunto com o *grep*.

Para saber qual é o PID do vim, por exemplo, um usuário poderia executar *ps aux | grep -i vim*. Antes de de executá-lo, no entanto, vamos entender o que faz cada parte desse comando: as opções *aux* garantem que o ps exiba processos de todos os usuários (*a*), o nome do usuário responsável pelo processo (*u*) e também aqueles processos que não estão, necessariamente, sendo executados naquele terminal (*x*). A barra vertical, ou pipe (|), faz com que o resultado seja direcionado para o comando *grep* que, por sua vez filtrará apenas as linhas que tenham a palavra *vim*.

3. KILL

Se um software travou ou precisa ser interrompido de qualquer forma, o *kill* é a solução. Basta executar o comando seguido do PID do processo para que a aplicação "morra". Se mesmo depois disso você perceber que o processo ainda existe, tente acrescentar a opção *-9* ao comando: *kill -9 PID*. Assim você força o processo a ser interrompido a qualquer custo.

Se quiser matar mais de um processo ao mesmo tempo, basta listar os PIDs separando-os com um espaço, logo depois do comando kill. Exemplo: *kill 3657 6785 3456*.

4. KILLALL & PKILL

Caso o usuário prefira, ele também pode matar de uma vez só todos os comandos selecionado ao nome de um programa. Para isso, basta usar o comando *killall* seguido do nome do software em questão, como *killall vim*.
Porém, o *killall* exige uma certa rigidez ao informar o nome do processo. Caso o usuário não tenha certeza do nome completo, pode tentar o *pkill*, que faz diversas associações com a palavra-chave digitada.

5. RENICE

Todos os processos do Linux possuem prioridades de execução, variando em uma escala que vai de 19 (menos significativa) a -20 (mais significativa).

Por padrão, os processos executados por um usuário ganham a prioridade 0, mas por meio do comando *renice* é possível alterar esse valor para algum nível entre 0 e 19. Apenas o usuário administrador (*root*) é capaz de ir além, alterando prioridades de qualquer processo e chegando até o nível máximo de -20.

Para realizar esse tipo de operação, basta seguir a sintaxe *renice novaprioridade -p PID*. Se quiséssemos dar mais prioridade a um processo de PID 1516, por exemplo, usaríamos: *sudo renice -10 -p 1516*. Lembre-se que o sudo exigirá a senha do seu usuário antes de executar o comando.

5. (Visando o Projeto final 1) Implemente o jogo Snake, disponível no vídeo:

O código do projeto estára no github portanto veja abaixo um print da simulação de como o game funciona

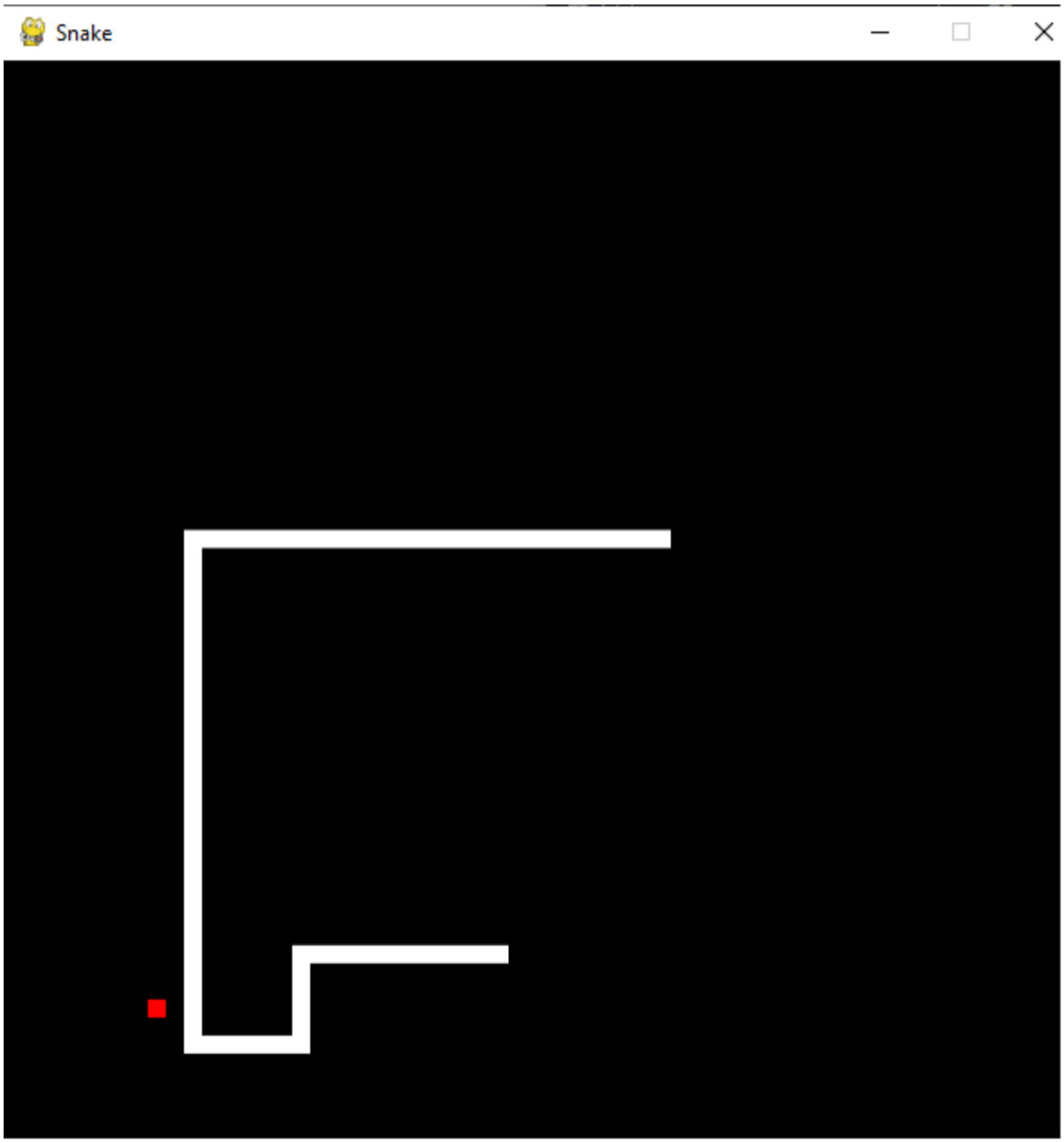

Figura 1: Snake game em python utilizando a pygame.

-