Exercícios SOA B1

1). Proteção de Memória: é responsável pela manutenção da integridade dos dados de um processo quando este compartilha um mesmo computador com outros processos. Ela é implementada em conjunto pelo hardware e sistema operacional, que disponibilizam para cada processo um espaço de memória exclusivo.

2). Multiprogramação com partições fixas: é usada em sistemas embarcados simples. Muitos dos programas modernos permitem que múltiplos processos executem simultaneamente, ou seja, quando um processo for bloqueado, outro poderá usar a CPU aumentando a sua utilização. Ao chegar no sistema, um job é colocado em uma fila de entrada juntamente associada à menor partição existente, **porém que seja grande o suficiente para armazena-lo e como, o tamanho dessas devidas partições são fixas, todo espaço que não é usado pelo job na partição será perdido**. Quando jobs estão chegando torna-se evidente quando a fila para uma grande partição está vazia, mas a fila para uma pequena partição está cheia, **nesse caso os jobs pequenos terão de esperar para que a memória seja liberada, mesmo que exista memória disponível**.

3). Os processos não enxergam a memória física, hardware usado para endereçar os circuitos integrados de memória, e sim a memória lógica, que é a memória capaz de ser endereçada e acessada pelo conjunto de instruções do processador, onde cada processo possui a sua memória lógica que é independente da memória lógica dos outros processos.

Memória Física: é implementada pelos circuitos integrados de memória, pela eletrônica do computador. Ela possui espaço de endereçamento físico que é o conjunto formado por todos os endereços dos circuitos integrados que formam a memória.

Memória Lógica: possui um espaço de endereçamento lógico, **maior que o espaço de endereçamento físico**, é formado por todos os endereços lógicos gerados pelo processo, sendo gerado pela CPU e único por processo.

4) A memória possui 3 segmentos:

Text/code segment: onde o código do programa reside.

Stack segment: espaço onde variáveis automáticas que estão dentro de funções serão alocadas. Todas as variáveis declaradas no main () entram no stack. (***Variáveis locais***).

Heap segment: é um espaço mais estável de armazenamento de dados de memória. Memória alocada no Heap ficará lá até enquanto o programa durar. Sendo assim variáveis globais e estáticas são alocadas no Heap. (***Variáveis globais***).

5) Mapa de Bits: é uma forma simples de controlar alocação da memória, pois seu tamanho só depende do tamanho da memória e do tamanho da unidade de alocação. Cada bit do mapa representa uma unidade de alocação, sendo que se o bit for 0, a unidade está livre, caso contrário a unidade está ocupada. **Quanto menor for a unidade de alocação, maior será o mapa de bits e vice-versa**. O principal problema deste método ocorre quando for necessário trazer para a memória um processo que ocupa *k* unidades

de alocação. O gerente de memória deve procurar por *k* bits 0 consecutivos no mapa. Esta procura é excessivamente lenta, de forma que, na prática, os mapas de bit raramente são usados.

6) Listas Encadeadas: contém os segmentos de memória livres e encadeados. Uma possível configuração seria manter, em cada entrada, o endereço em que inicia, o seu comprimento e, evidentemente o ponteiro para a próxima entrada. A principal vantagem é que a atualização é simples e direta.

7) Tabela de Páginas Invertidas: cada entrada representa uma moldura de páginas ao invés de um endereço virtual. Desta forma, a entrada deve monitorar qual página virtual está associada àquela moldura de páginas. Economizam quantidade significativa de espaço.

8) TLBs (Translation Lookaside Buffers): é um dispositivo de hardware cujo propósito é mapear endereços virtuais em endereços físicos sem passar pela tabela de páginas. Ela constitui-se de um banco de registradores que armazenam um pequeno número de entradas, muito rápidas, contendo as tabelas de páginas mais utilizadas. Melhora bastante o desempenho no acesso à tabela de páginas, visto que registradores são muito mais rápidos que a memória RAM.

10) O sistema de arquivos de arquivos de Linux é uma grande árvore enraizada e mesmo assim temos sistema de arquivos em diferentes dispositivos. O sistema de arquivos enraizados é montado como parte do processo de inicialização. Cada um dos outros sistemas de arquivos que

for criado pelo usuário não será utilizável pelo sistema Linux até que esteja montando em um ponto de montagem. Um ponto de montagem é simplesmente um diretório no qual o sistema de arquivos em um dispositivo é inserido na árvore. Montagem é o processo de tornar o sistema de arquivos no dispositivo acessível.

11). Swap: é uma técnica criada na tentativa de melhorar o problema da insuficiência de memória durante a execução de alguns processos em ambientes multiprogramados. Essa técnica consiste em transferir automaticamente todo o processo da memória principal para o disco e vice-versa.

12) */proc:* é um diretório virtual que serve como "diagnóstico" e configurações em tempo real do kernel. É um diretório especial onde ficam todas as informações de depuração do kernel. Também se encontram algumas configurações que habilitam e desabilitam o suporte à alguma coisa do kernel.

13) a) */proc/cpuinfo:* mostra informações sobre o processador.

b) */proc/interrupts:* relaciona o número de interrupções por CPU por dispositivo de E/S. Ele exibe o número de IRQ, o número daquela interrupção manipulada por cada núcleo da CPU, o tipo de interrupção, e uma lista delimitada por vírgulas de motoristas que estão registrados para receber essa interrupção

c) */proc/stat:* diversas informações sobre a atividade do kernel estão disponíveis no arquivo /proc/stat. Todos os

números relatados neste arquivo são agregados desde o primeiro sistema inicializado.

14) a) $*: retorno aos argumentos.

b) $#: total de argumentos que foram passados.

c) $0: nome do script executado.

d) $$: número do PID (process ID).

15) • cat /proc/cpuinfo: informações específicas.

• free –h: obter informações sobre o uso e disponibilidade da memória no Linux.

• cat /etc/hostname: nome dado ao sistema.

• sudo dmidecode | more: mostra uma maior quantidade de informações pausadamente.

16)

```
cd /var/log
ls *.log | grep Xorg | xargs sudo rm
```

cd: serve para acessar e mudar de diretório corrente. Ele é utilizado para a navegação entre pastas.

/var: contém arquivos que são modificados com o decorrer do uso do sistema (e-mail, temporários, filas de impressão, manuais).

/var/log: **arquivos de log do sistema (erro, logins, etc....).**

grep: procurar texto em uma string ou dentro de arquivos e mostrar linhas, ocorrências, usar combinações para pesquisar e o resultado da pesquisa ser mostrado na tela.

Xorg: é que fornece uma interface entre seu hardware e o software gráfico que você quer rodar.

Xargs: é utilizado para construir e executar linhas de comando Linux. Pode ser utilizado para efetuar uma divisão dessa lista em sub-listas menores, e passar os argumentos ao comando requisitado, em partes.

| (pipe): envia a saída de um comando para a entrada do próximo comando para continuidade do processamento.

17)

```
cd -
date > hoje.txt
cat hoje.txt
```

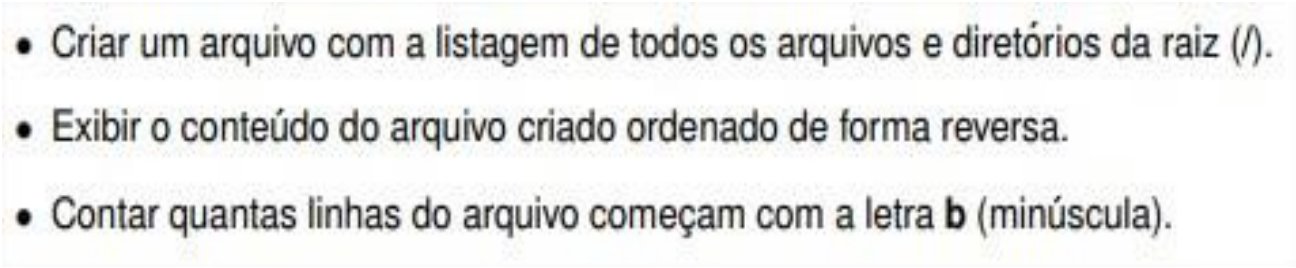
date: data e hora do sistema.

>: saída padrão.

cat: imprime na tela o conteúdo do arquivo.

18) SYSFS: É descrito como a união dos sistemas de arquivo proc, devfs e devpty. O sistema de arquivos sysfs enumera os dispositivos e canais conectados pelo espaço do usuário. É desenvolvido para lidar com as opções específicas do driver e do dispositivo, que antes estavam localizadas no /proc/, e incluir a adição dinâmica de dispositivos previamente oferecidas pelo devfs.

19)



•. LS – L > ARQUIVO.TXT

•. CAT ARQUIVO.TXT | SORT –R

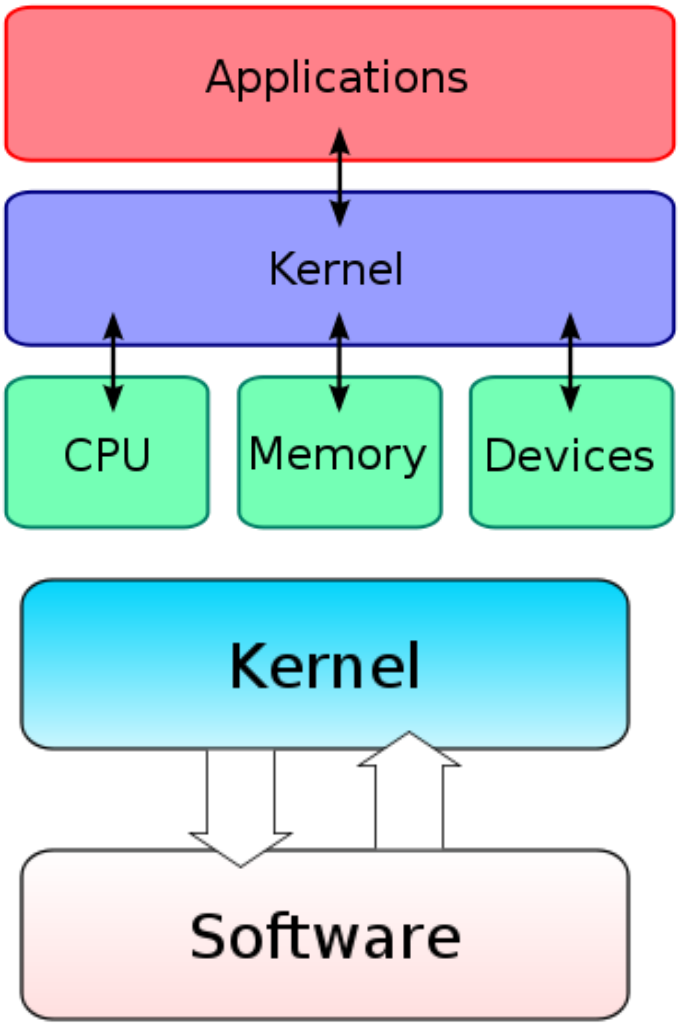
•. CAT ARQUIVO.TXT | WC –L | GREP "^C.*"

20) lscpu: obter informações sobre o processador.

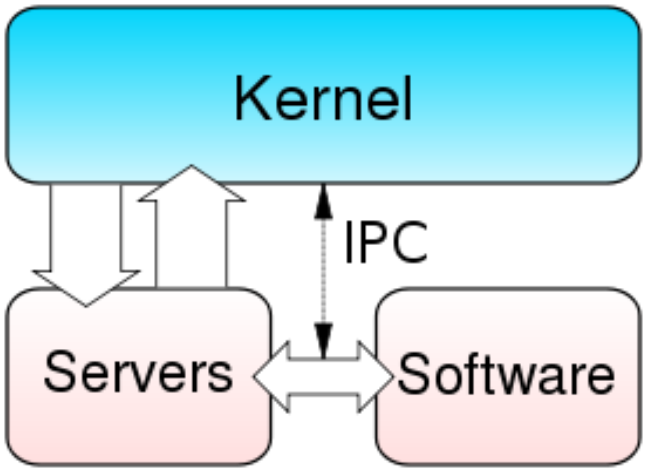
Lshw –short: listar o hardware.

Lsusb: dispositivos usb.

21) Kernel Monolítica: é uma arquitetura de núcleo onde todo o núcleo é executado no espaço de núcleo. Ou seja, é um kernel que possui todos os códigos de suporte necessários. O núcleo define uma camada de alto nível de abstração sobre o hardware do computador, com um conjunto de primitivas ou **chamadas de sistema** para implementar os serviços do sistema operacional como gerenciamento de processos, concorrência e gestão de memória em um ou mais módulos.



22) Microkernel: é uma arquitetura de núcleo de um sistema operativo cujas funcionalidades são quase todas executadas fora do núcleo, em oposição a um núcleo monolítico. Os processos se comunicam com um núcleo mínimo, usando o "**espaço do sistema**". Neste local os aplicativos têm acesso a todas as instruções e a todo o hardware e deixando o máximo de recursos rodando no "**espaço de usuário**" em que o software tem algumas restrições, não podendo acessar hardwares, nem tem acesso a todas as instruções.



23)

a) > e >>

b) <

c) |

d) 2 >

e) 2 > &1

a) >: saída padrão.

>>: redireciona a saída de um programa/ comando para algum dispositivo ou adiciona as linhas ao final do arquivo ao invés do dispositivo de saída padrão.

b) <: entrada.

c) | (pipe): envia a saída de um comando para a entrada do próximo comando para continuidade do processamento

d) 2>: saída de erro padrão.

e) 2>&1: a saída de erro está sendo redirecionada, mas não para um arquivo, e sim para outra stream. O uso do & indica que a saída irá para outro file descriptor, e não para um arquivo.

24). Toda saída gerada por um comando no bash, é direcionada para o output padrão, no caso o terminal, podemos trabalhar com a saída do comando direcionando-a para outro comando ou não direcionando para nenhum local (null), entre outras opções.

25) PATH: é uma variável do sistema usada pelo sistema operacional para localizar executáveis necessários da linha de comando ou da janela Terminal.

26) PS1: altera o prompt de comando de acordo com o sistema operacional.

27) Systemd: é um sistema de init usado em distribuições Linux para inicializar o espaço do usuário e gerenciar todos os processos posteriormente, em vez dos sistemas de inicialização UNIX.

28) Bootloader: é um programa que entra em ação sempre que iniciamos o computador ativando o sistema operacional certo. Isto serve para qualquer dispositivo dotado de sistema operacional: tablets, smartphones e PCs.

29) Bios: é um tipo de firmware (conjunto de instruções operacionais programas diretamente no hardware de um equipamento) usado para realizar a inicialização do hardware durante o processo de inicialização, e para

fornecer serviços de tempo de execução para sistemas operacionais e programas.

30) O primeiro estágio é um programa maior que o código no MBR ele se encontra no resto dos setores da primeira lista, por ser limitado a um pequeno número de setores tem tendência a trabalhar com um único sistema de arquivos, mas é o suficiente para carregar um sistema operacional ou o segundo estágio que por ser um programa maior pode trabalhar com vários sistemas de arquivos e sistemas operacionais. O pequeno programa normalmente não é o sistema operacional, mas apenas um segundo estágio do sistema de inicialização, assim como o Lilo ou o Grub. Ele será então capaz de carregar o sistema operacional apropriado, e deve carregar drivers de dispositivos e outros programas que são necessários para a operação normal de um sistema operacional. O processo é considerado completo quando o computador está pronto para ser operado pelo usuário.

31) Exercício Repetido

32) Exercício Repetido

33) Tirado da primeira lista feita em sala de aula

DATE | TEE –A /TMP/HOJE.TXT

tee: permite que a saída de um comando seja gravada em um arquivo ao mesmo tempo em que é exibida na tela.

(-a): indica que a saída do comando deve ser acrescida ao conteúdo do arquivo especificado.

34) Exercício Repetido

35) Tirado da primeira lista feita em sala de aula

1. Exiba o conteúdo do arquivo /etc/fstab
2. Conte o número de linhas do arquivo /etc/fstab
3. Exiba somente as duas primeiras linhas do arquivos /etc/fstab (Dica: man head)
4. Exiba somente as duas ultimas linhas do arquivos /etc/fstab (Dica: man tail)
5. Execute os seguintes comandos no diretório padrão (*home*) do usuário:
   seq 1 10 > f1
   seq 10 -1 1 > f2
   Os arquivos f1 e f2 serão criados. Ordene o arquivo f1. Ordene novamente agora utilizando a opção -g.
6. Exiba a junção dos arquivos f1 e f2

1. CAT /ETC/FSTAB.

2. CAT /ETC/FSTAB | WC –L.

3. CAT /ETC/FSTAB | HEAD –N 2.

4. CAT /ETC/FSTAB | TAIL –N 2.

5. CAT F1 | SORT.

CAT F1 | SORT –G.

6. CAT F1 | F2.

36) Exercício Repetido

37) Exercício Repetido

38) Exercício Repetido

39) Exercício Repetido

40) Exercício Repetido