3ª Prova de Sistemas Operacionais – 2017/1

Aluno: ……

1- (0,5) É possível criar um *link* simbólico para um arquivo que não existe? E um *hard link*? Justifique sua resposta.

2- (1,0) Considere um sistema de arquivos UNIX com organização do tipo i-node cujo mapa de disco contém 13 ponteiros para blocos de disco: 10 diretos, 1 indireto (*single indirect block*), 1 duplamente indireto (*double indirec block*) e 1 triplamente indireto (*triple indirect block*). Assuma que cada bloco do disco é de 4KB (4096 bytes) e que um ponteiro para um bloco do disco ocupa 4 bytes. Quantas operações de disco são requeridas para ler o byte de número 100.000 de um arquivo? Assuma que o i-node já esteja na memória. Explique a sua resposta.

3- (1,0) Quantas operações em disco são necessárias para acessar o terceiro bloco do arquivo /home/joao/so/projeto.c, considerando i-nodes que só têm endereços diretos de blocos? Suponha que o diretório raiz esteja na memória, mas nenhum outro componente ao longo do caminho se encontra na memória. Suponha também que todos os diretórios caibam em um único bloco de disco. Explique as suas respostas.

4- (0,5) Uma máquina tem endereços virtuais de 32 bits e uma memória física com endereços de 20 bits. Páginas têm tamanho 4KB. Quantas posições são necessárias para a tabela de páginas em um esquema com apenas um nível de paginação? E em uma tabela de página invertida?

5- (1,0) Se o algoritmo FIFO de substituição de páginas for usado em um sistema com 4 páginas de MP e 8 de memória virtual, quantas faltas de página serão geradas com a seqüência de referências: 0 1 7 2 3 2 7 1 0 3. Considere que as 4 páginas da MP estão inicialmente vazias. Repita para o algoritmo LRU.

6- (2,0) Considere um sistema com páginas de 4Kbytes, endereçamento lógico de 16 páginas, endereçamento físico de 8 frames, e as tabelas de páginas dos processos em execução dadas abaixo. a) Explique em qual endereço físico a MMU traduz o endereço lógico 13K (para os dois processos); b) Considere que o processo A utilize apenas as seguintes partes espaço do endereçamento virtual: 0 a 20K-1, para endereços utilizados nos segmentos código e dados; e 58K a 64K-1, para endereços utilizados no segmento de pilha. Construa para o Processo A uma tabela de páginas dois níveis, com 4 entradas no primeiro nível.

	Processo A		Processo B	
	Bit de Valid.	Número da moldura	Bit de Valid.	Número da moldura
0	1	0	1	7
1	0	6	0	1
2	0	-	0	2
3	1	4	0	-
4	1	3	1	5
5	0	-	0	1
6	0	-	0	4
7	0	-	0	3
8	0	-	0	-
9	0	-	0	5
10	0	-	0	2
11	0	-	0	-
12	0	-	0	4
13	0	-	0	-
14	1	1	0	-
15	0	-	1	2

7- (1,0) Com relação à gerência de memória virtual, fale sobre o Conjunto de Trabalho (Working Set).

8- (1,0) Considere as seguintes sentenças em relação à técnica de IPC FIFO. Justifique a(s) alternativa(s) incorreta(s).

I. FIFO permite criar um canal de comunicação entre processos que não possuem ancestral comum.

II. FIFO cria um arquivo especial no sistema de arquivos, do tipo *pipe*, que é removido quando o processo que criou o FIFO é finalizado.

III. FIFOs também podem ser criados via shell com o comando *$* ***mkfifo*** *<meuFIFO>*.

9- (1,0) Escreva um programa em que o processo pai envia a mensagem "Bom dia, meu filho!" para o processo filho. A comunicação entre os processos deve ser feita através de um *pipe*.

10- (1,0) Suponha que um segmento de memória compartilhada tenha sido criado através de um programa X. Comente os programas test_shm1 e test_shm2 abaixo preenchendo os locais de comentários e os conteúdos dos printf's.

```
/* test_shm1: este programa ********************************************** */

#include …
#define KEY 123
#define MSG "Bom dia!"

int main() {
      int shmid ; /* ***************************************************** */
      int size = 1024 ;
      char path="*****************************************************" ;
      char *mem ;
      int flag = 0;

      /* ************************************************************** */
      if ((shmid = shmget(ftok(path,(key_t)KEY), size,0)) == -1)
            { perror("Erro no shmget") ;exit(1) ; }

      printf("**********************: %d \n",getpid()) ;
      printf("**********************: %d \n",shmid) ;
      printf("**********************: %d\n",ftok(path,(key_t)KEY)) ;

/* ********************************************************************** */
      if ((mem = shmat (shmid, 0, flag)) == (char*)-1){
            perror("***************************") ;
            exit (1) ; }

/* ********************************************************************** */
      strcpy(mem,MSG);
      exit(0);
}


/* test_shm2: este programa ********************************************** */

#include …
#define KEY 123
int main() {
      int shmid ; /* ***************************************************** */
      int size = 1000 ;
      char path="*****************************************************" ;
      char *mem ;
      int flag = 0 ;


      /* ************************************************************** */
      if (( shmid = shmget(ftok(path,(key_t)KEY), size,0)) == -1) {
            perror("Erro no shmget") ;
            exit(1) ; }
      printf("****************************************: %d \n",getpid()) ;
      printf("****************************************: %d \n",shmid) ;
      printf("******************************: %d\n", ftok(path,(key_t)KEY));

      /* ************************************************************** */
      if ((mem = shmat (shmid, 0, flag)) == (char*)-1){
            perror("******************************") ;
            exit (1) ;}

      /* ************************************************************** */
      printf("\t==>%s\n",mem) ;
      exit(0);
}
```

BOA PROVA.