3ª Prova de Sistemas Operacionais – 2013/2 – Prof. José Gonçalves

Aluno: .....

1. **(1,0)** Considere os programas abaixo. Se `execl` tem sucesso, qual é a saída do programa exec4? Se `execl` falha, qual é a saída de exec4? Justifique as suas respostas.

```
/* Programa exec4 */
...
int main() {
    pid_t pid;
    pid = fork();
    if(pid == 0)
    {
        printf("Child runs\n");
        execl("atest", "atest", "dog", 0);
        printf("Child done\n");
    }
    else {
        wait(NULL);
        printf("Parent done\n");
    }
}
```

```
/* Programa atest */
...
int main(int argc, char * argv[]) {
    printf("Got a [%s]", argv[1]);
}
```

2. **(1,0)** Dada a seguinte parte de um programa, quantas vezes é impressa a frase "Prova Final."? Justifique montando a árvore de processos.

```
main()
{
    ...
    y=0;
    for (i=0; i<2; i++) {
        x=fork();
        if (x==0)
            y=fork();
        if (y==0) {
            fork();
            printf("Prova Final.\n");
        }
    }
    ...
}
```

3. **(1,5)** Explique o que faz este programa?

```
int count=0;
main()
{
   char c='x';
   if (pipe(p) < 0)
      error("pipe call");
   signal(SIGALRM,alarm_action);
   for(;;) {
      alarm(20);
      write(p[1],&c,1);
      alarm(0);
      if((++count%1024)==0)
         printf("%d chars in pipe\n, count");
   }
}

alarm_action()
{
   printf("write blocked after %d chars \n", count);
   exit(0)
}
```

4. **(1,0)** Com relação ao POSIX *pthreads*, assinale V (Verdadeiro) ou F (Falso). Justifique as alternativas falsas.

   pthread_create: cria uma nova *thread* e a faz "executável".

   pthread_join: suspende a execução de outras *threads* até que a *thread* explicitada em *join* termine sua execução. É um dispositivo para sincronização de *threads.*

pthread_mutex_lock: utilizada por uma *thread* para adquirir exclusividade de manipulação sobre uma variável mutex específica. Se esta variável já estiver sendo utilizada, a *thread* que invocou o lock é bloqueada até que a variável mutex seja liberada;
pthread_cond_t: tipo de dados para criação de variáveis de condição, utilizadas em sincronização de *threads.*
pthread_donothing: suspende a execução da *thread*.

5. **(1,0)** O que faz o trecho de programa abaixo?

```
...
sigset_t newmask, oldmask;
...
sigemptyset(&newmask);
sigaddset(&newmask, SIGINT);
sigprocmask(SIG_BLOCK, &newmask, &oldmask);
...                        /* escreva aqui um comentário */
sigprocmask(SIG_SETMASK, &oldmask, NULL);
pause();
```

6. **(1,5)** Considere as seguintes sentenças em relação à técnica de IPC FIFO. Justifique as alternativas incorretas, caso existam:
   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*>.

7. **(1,0)** Considere o seguinte array: var A: array[1..100] of array[1..100] of integer; onde A[1][1] está na posição 200 em um sistema com memória paginada com páginas de tamanho 200. O processo que manipula a matriz está na página 0 (0-199), desta forma as instruções são buscadas na página 0. Considerando 3 molduras (frames), quantas *page faults* são geradas pelo seguinte código de inicialização do array, usando LRU, sabendo-se que a moldura 1 tem o código de inicialização e as outras duas molduras estão vazias inicialmente. Explique a sua resposta.

```
for j:= 1 to 100 do
   for i:= 1 to 100 do
       A[i][j]:=0;
```

8. **(2,0)** Considere um sistema com páginas de 4Kbytes, endereçamento lógico de 16 páginas, e endereçamento físico de 8 frames. Considere as seguintes tabelas de páginas dos processos em execução:
   a) Explique em qual endereço físico a MMU traduz o endereço lógico 12K (para os dois processos)
   b) Considere que o processo A utilize apenas partes espaço do endereçamento virtual, sendo essas:
      - 0 a 20K-1: endereços utilizados nos segmentos código e dados
      - 58K a 64K-1: endereços utilizados no segmento pilha

   Construa para o Processo A uma tabela de páginas dois níveis (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

BOA SORTE.