2ª. Prova de Sistemas Operacionais – 2017/1 – Prof. José Gonçalves

Aluno: .....

1. Qual é a saída do seguinte programa?

```
int main()
{
  int pid, x = 4;
  pid = fork();
  if ( 0 == pid ) {
    fork();
    x=x+2;
  }
  else { x--; }
  printf("x=%d\n",x);
}
```

2. Considere o programa abaixo:
   a) Quantos processo serão criados?
   b) Desenhe uma árvore que descreva os processo criados.
   c) Altere o programa de modo que apenas sejam gerados 3 filhos.
   d) Altere o programa de modo que o pai espere pelo fim de cada um dos filhos.
   e) Altere o programa de modo a que o pai apenas espere pelo segundo filho, mas sem bloquear.

```
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>

void main(void)
{
   pid_t pid;
   int f;
   for (f = 0; f < 3; f++)
   {      pid = fork();                                  /* Cria um processo */
          if (pid > 0) printf("Pai: Eu sou o PAI\n");    /* Código do pai */
          else  sleep(1);                                /* Código do filho */
   }
}
```

3. Considere o seguinte programa.

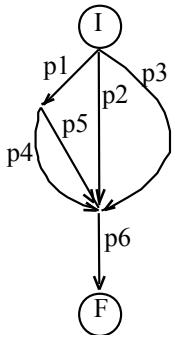
```
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>

void main(void)
{
    pid_t pid;
    int f;

    fork();
    printf("1\n");
    fork();
    printf("2\n");
    fork();
    printf("3\n");
}
```

   (a) Desenhe uma árvore que descreva o conjunto dos processos criados.
   (b) É possível que o número 1 apareça depois do 3? Se sim, explique o porquê (mostre um exemplo).

4. Use semáforos para sincronizar as operações dos processos p1, p2, ..., p6, conforme com o grafo a seguir.



5. Por qual motivo um usuário daria preferência à utilização de threads ao invés de processos em um programa concorrente?

6. Use primitivas de mutexes da biblioteca *Pthreads* para prevenir inconsistência de dados devido a operações de múltiplas threads sobre a função *functionC()* abaixo.

```
1 int counter=0;
2 /* Function C */
3 void functionC()
4 {
5    counter++
6 }
```

7. Implemente um programa que, após criar um novo processo, tem o seguinte comportamento cíclico:
- o processo pai aguarda 2 segundos e envia ao filho um sinal SIGUSR1;
- o processo filho, após a recepção do sinal, imprime "Apanhei o sinal SIGUSR1".

8. Explique o funcionamento do seguinte programa:

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>

void funcao() {
   execlp("ls", "ls", NULL);
}

void main()
{
   pid_t pid;
   int i;

   pid = fork();
   if (pid == 0)
        for (i=0; i<3; i++)
        {
             kill (getppid(), SIGUSR1);
             sleep(5);
        }
   else if (pid >0)
        {
             signal (SIGUSR1,funcao);
             for (; ;)
                  pause();
        }
}
```

9. Explique o funcionamento do seguinte programa:

```
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>

int main(void)
{
 pid_t pid;
 int aux;
 int status;

 pid=fork();
 if (pid<0)
 {
     perror("Erro ao cria o processo\n");

     exit(-1);
 }
 else
 {
     if (pid > 0) /* Código do Pai */
     {
          printf("Pai\n");
          do
          {
               aux = waitpid(pid, &status, WNOHANG);
               if (aux==-1)
               {
                    perror("Erro em waitpid");
                    exit(-1);
               }
               if (aux == 0)
               {
                    printf(".\n");
                    sleep(1);
               }
          } while (aux == 0);
          if (WIFEXITED(status))
          {
               printf("Pai: o filho retornou o valor:%d\n", WEXITSTATUS(status));
          }
     }
     else /* Código do filho */
     {
          printf("Filho\n");
          sleep(5);
          printf("Filho a sair\n");
          exit(5);
     }
     exit(0);
 }
}
```

BOA PROVA.