2ª Prova de Sistemas Operacionais – 2013/2

Aluno: .....

1. (1,5) Boa parte dos sistemas operacionais atuais (incluindo Linux) apresenta uma associação fixa (binding) entre uma ULT (User Level Thread) e uma KLT (Kernel Level Thread), seguindo o modelo "um para um". Já algumas versões mais antigas de UNIX permitem ao processo ter uma quantidade x de ULTs e y de KLTs, sendo que, dinamicamente, qualquer uma das ULTs pode ser binded a qualquer uma das KLTs desse processo. Quais são as vantagens e disvantagens desse modelo, se comparado com o modelo "um para um"

2. (2,0) Considere o seguinte programa (assuma que não existem erros na execução das chamadas):

```
int main() {
      int i=0;
      while (i<10) {
                i = i + 2;
                if (fork() != 0) {
                      printf ("I'm %d\n", i);
                      i = i - 1;
                      exit(0);
                }
                sleep(5);
      }
      return 0;
}
```

a) Explique o seu funcionamento, justificando qual é o resultado da execução do programa (mostre o que é impresso e o porquê). Assuma que o *quantum* de cada processo é muito inferior a 5 segundos.
b) Diga se existe a formação de processos *zombies* e/ou criação de órfãos com adoção por parte do processo *init*.

3. (1,5) Dada a seguinte parte de um programa, quantas vezes serão executados os programas **exame** e **alunos**? Justifique.

```
...
pid = fork();
pid = fork();
for(i=0; i<5; i++)
{
pid = fork();
        execlp ("exame", "exame", 0);
        if (pid == 0)
                break;
}
execlp("alunos", "alunos","SO",0);
...
```

4. (1,5) Considere o seguinte programa:
…

```
void funcao() {

    execlp("prog", "prog", 0);

}

void main() {

   pid_t pid;

   int i;

   pid = fork();

   if (pid == 0)
```

```
      for(i=0; i<3; i++)  {
         kill(getppid(), SIGUSR1);
         sleep(2);
      }
      else
     if (pid >0) {
         signal(SIGUSR1,funcao);
            for(; ;)
            pause();
      }
}
```

Quantas vezes é executado o programa **prog**? Justifique.

5. (2,0) Considere uma extensão do problema do produtor-consumidor definida da seguinte forma:
a) Há dois processos consumidores e um processo produtor. Os processos executam concorrente e assincronamente.
b) O produtor não pode escrever no buffer até que ambos os consumidores tenham lido dele. É aberta uma exceção para a primeira escrita.
c) Nenhum consumidor pode ler o conteúdo de um buffer indefinido.
d) Nenhum consumidor pode ler o mesmo conteúdo duas vezes sucessivamente.
e) Ambos os consumidores podem executar seus eventos simultaneamente.

Defina semáforos e use primitivas P e V para impor a exclusão mútua e sincronizar os processos.

Processo Produtor	Processo Consumidor1	Processo Consumidor2
...	...	...
repeat	repeat	repeat
...	…	...
coloca dados no buffer	retira dados no buffer	retira dados no buffer
...	…	...
until forever	until forever	until forever

6. (1,5) Quais os problemas com a solução que desabilita as interrupções para implementar a exclusão mútua?