UFES - DEPARTAMENTO DE INFORMÁTICA

2ª. Prova de Sistemas Operacionais - Período: 2013/1 - Profª. Roberta Lima Gomes

1) **(2,0)** No primeiro trabalho prático da disciplina, sobre o tratamento de sinais, foi solicitado que uma vez que um filho (que estava em foreground) da bgsh é passado para o estado Suspenso, ele deve "ele não poderá continuar a executar, mesmo que ele receba um sinal SIGCONT". Descreva como o seu grupo implementou esse comportamento, explicando porque foi implementado dessa forma e se ocorreram problemas durante a execução.

2) **(3,0)** Considere que existe um serviço com 3 guichets de atendimento (três processos servidores) ao qual chegam clientes (processos clientes). O sistema deve funcionar do seguinte modo:
   - Se existirem processos servidor livres o processo cliente deve ser atendido de imediato.
   - Se não existirem processos clientes à espera de atendimento os processos servidores devem ficar bloqueados.
   - O número máximo de clientes em espera é de K. Para além desse limite o serviço é recusado ao cliente.

//PROCESSO CLIENTE	**//PROCESSO SERVIDOR**

`for (;;){`
`  If(NovoCliente()) {`
`  Servico();`
`  } else ...`
`}`
	`for (;;){`
	`  ProximoServico();`
	`  Atendimento();`
	`}`

Programe as seguintes funções que os processos devem invocar: `ProximoServico()` e `NovoCliente()`. Utilize semáforos para sincronizar os processos. Programe em C ou pseudocódigo e defina as variáveis que necessitar.
Obs: Não se preocupe com os procedimentos `Atendimento()` e `Servico();` apenas se sabe que são procedimentos executados pelos processos Servidor e Cliente, respectivamenete, e que terminam ao fim de um intervalo de tempo finito.

3) **(2,5)** Explique o que são *Threads*, falando igualmente sobre *User Level Threads* (ULT) e *Kernel Level Threads* (KLT), ressaltando suas vantagens e desvantagens.

[VIRE A FOLHA!]

4) **(2,5)** Considere o programa abaixo:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
/* para a função ftok */
#include <sys/types.h>
#include <sys/ipc.h>

int main(int argc, char *argv[]) {

   // ID da memória

   int mem_ID;

   //acesso a nossa memória

   int *minhaMemoria;

   // valor da chave

   key_t chave;

   chave = ftok( "./teste.c", 5);

   if(chave == -1) { perror("Erro ao criar a chave"); exit(1);}

   printf( "Valor da chave: %#x\n", chave );

   // syntaxe do shmget:

   // int shmget(key_t key, size_t size, int shmflags);

   mem_ID = shmget( chave, sizeof(int), 0666|IPC_CREAT|IPC_EXCL );

   if(mem_ID == -1) {perror("Erro ao criar memoria");exit(2);}

   printf( "ID da memória: %d\n", mem_ID );

   // syntaxe do shmat:

   // int shmat(int shmid, void *addrAttachment, int flag);

   minhaMemoria = (int*) shmat (mem_ID , (void *)0, 0);

   if ((int)minhaMemoria == -1){ perror("attachment impossivel"); exit(3);}

   *minhaMemoria = rand() % 100;

}
```

a) Explique o que ele faz.
b) Há ocorrência de IPC (*Interprocess communication*)? Por quê?
c) O que aconteceria se você executasse esse programa duas vezes seguidas?

Boa Prova !!!