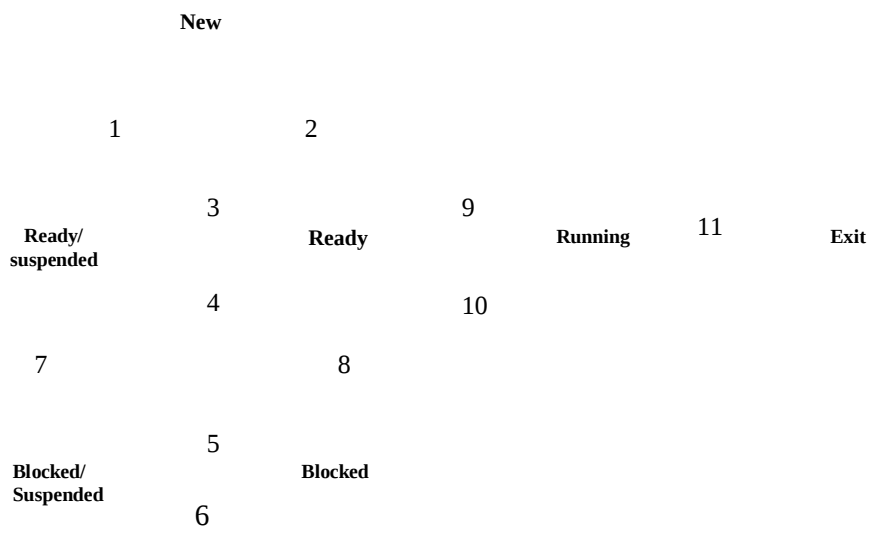
UFES - DEPARTAMENTO DE INFORMÁTICA

2ª. Prova de Sistemas Operacionais / Sistemas de Programação II

Período: 2010/2
Profa. Roberta Lima Gomes

ALUNO: _____

1) Dado o diagrama de estados de processos abaixo, descreva e explique os eventos que provocam cada uma das transições entre estados para. **(2,5)**



2) Assumindo que todas as operações *fork()* retornarão com sucesso, quantos processos serão criados a partir do momento que o usuário solicita ao S.O a execução do seguinte programa? O que será impresso? **(2,5)**

```
main()
{
      int i=1;
      int ret_val;

      while(i <= 3)
      {
            if ((ret_val = fork()) == 0) {
                  printf("In child %d. \n", i);
                  i = i + 1;
            } else {
            printf("In parent %d. \n", i);
            exit(0);
            }
      }
}
```

3) Considere o problema dos leitores e escritores, onde existem diversos processos que eventualmente fazem acessos de leitura a uma base de dados e diversos processos que eventualmente fazem acessos de escrita à mesma base. Vários acessos de leitura podem ocorrer simultaneamente, mas um acesso de escrita não pode ocorrer simultaneamente com acessos de nenhum tipo. Considere o código a seguir para os processos de leitura (leitor) e de escrita (escritor). Suponha que todos os semáforos são iniciados com valor 1:

```
rc       //Número de leitores
wc       //Número de escritores, apenas um escritor de cada vez pode ter acesso aos
         //dados compartilhados
mutex_rc// Protege o acesso à variável rc
mutex_wc//Protege o acesso à variável wc
mutex    //Impede que + do que 1 leitor tente entrar na região crítica
w_db     //Indica a um escritor se este pode ter acesso aos dados
r_db     //Permite que um processo leitor tente entrar na sua região crítica
```

Inicialização

```
rc = 0
wc = 0

//semáforos
mutex_rc = 1
mutex_wc = 1
mutex = 1
w_db = 1
r_db = 1
```

Escritor

```
while (TRUE){
  down(mutex_wc);
  wc++;
  if (wc == 1)
    down(r_db);
  up(mutex_wc);
  down(w_db)
  …
  //Escrita
  …
  up(w_db)
  down(mutex_wc);
  wc--;
  if (wc == 0)
    up(r_db);
  up(mutex_wc);
}
```

Leitor

```
while (TRUE){
  down(mutex);
  down(r_db);
  down(mutex_rc);
  rc++;
  if (rc == 1)
    down(w_db);
  up(mutex_rc);
  up(r_db);
  up(mutex);
  …
  //Leitura dos dados
  …
  down(mutex_rc);
  rc--;
  if (rc == 0)
    up(w_db);
  up(mutex_rc); )
```

(a) Essa solução deve dar prioridade a um tipo de processo. Qual? Leitores ou aos escritores? Explique sua resposta. **(1,5)**
(b) Explique a importância do semáforo mutex dando um exemplo de problema que poderia ocorrer caso as operações sobre ele fossem retiradas. **(1,0)**

4) "Existem n passageiros, que repetidamente aguardam para entrar em um carrinho da montanha russa, fazem o passeio, e voltam a aguardar. Vários passageiros podem entrar no carrinho ao mesmo tempo, pois este tem várias portas. A montanha russa tem somente um carrinho, onde cabem C passageiros (C < n). O carrinho só começa seu percurso se estiver lotado." Resolva usando semáforos para sincronizar os processos Passageiro e Carrinho. **(2,5)**

Boa Prova!