Sistemas de Operação

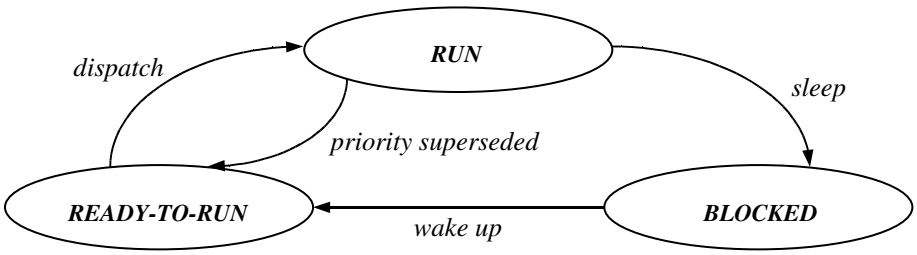
Ano lectivo de 2006-2007

Parte A (*10 valores*)

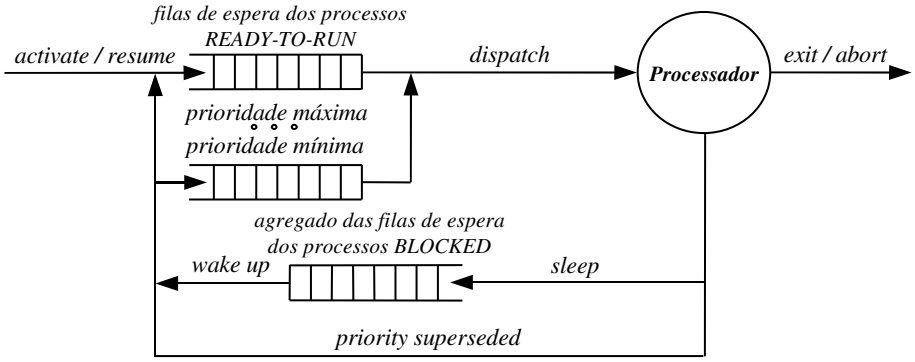
1. Indique quatro características comuns aos sistemas de operação actuais. Tome como referência na sua análise o Linux e o Windows XP da Microsoft.
2. Indique quais são as funções principais desempenhadas pelo *kernel* de um sistema de operação. Neste sentido, explique porque é que a sua operação pode ser considerada como um *serviço de excepções*.
3. Distinga as diferentes políticas de *prevenção de deadlock no sentido estrito*. Dê um exemplo ilustrativo de cada uma delas numa situação em que um grupo de processos usa um conjunto de blocos de disco para armazenamento temporário de informação.
4. Descreva detalhadamente o mecanismo de tradução de um *endereço lógico* num *endereço físico* numa organização de memória virtual paginada.
5. Uma característica comum aos sistemas de operação contemporâneos é possibilitarem a partilha de ficheiros. Descreva neste contexto a arquitectura base de um servidor de ficheiros e indique que vantagens e inconvenientes apresentam.

Parte B (*10 valores*)

A figura apresenta o diagrama de transição de estados do *scheduling* de baixo nível do processador



Suponha que foi implementada uma disciplina de *scheduling* com 8 níveis de prioridade (0- representando a prioridade máxima, 7 – representando a prioridade mínima), como a figura abaixo ilustra.



Assuma que a política de *scheduling* é de tipo *non-preemptive*, obedecendo às propriedades seguintes

- em circunstâncias normais, um processo no estado READY-TO-RUN é colocado na fila de espera correspondente ao seu nível de prioridade, nível esse que, em princípio, será mantido durante toda a sua vida;
- ocorre uma comutação de contexto, por *priority superseded*, sempre que um processo com um nível de prioridade superior àquele que detém presentemente o processador, passa ao estado READY-TO-RUN;
- nestas circunstâncias, ao processo que perde o processador é atribuído, como compensação, e apenas para o próximo agendamento de execução, uma prioridade cujo nível é uma unidade superior à actual, regressando depois ao nível da sua prioridade base.

Admita ainda que foram definidas as estruturas de dados seguintes

Entrada (*simplificada*) *da Tabela de Controlo de Processos*

```
typedef struct
        { BOOLEAN busy;             /* sinalização de entrada ocupada */
          unsigned int pid,              /* identificador do processo */
                       pstat,          /* estado do processo: 0 - RUN
                                         1 - BLOCKED 2 - READY-TO-RUN */
                       baseprior,       /* nível de prioridade base */
                       actualprior;    /* nível de prioridade actual */
          unsigned char intreg[K];         /* contexto do processador */
          unsigned long addspace,    /* endereço da região de memória
                           principal onde está localizado a descrição
                               do espaço de endereçamento do processo */
                        iospace;     /* endereço da região de memória
                           principal onde está localizado a descrição
                                     do contexto de I/O do processo */

        } PCT_ENTRY;
```

Nó de lista biligada

```
struct binode
      { unsigned int info;                        /* valor armazenado */
        struct binode *ant,            /* ponteiro para o nó anterior */
                      *next;           /* ponteiro para o nó seguinte */
      };

typedef struct binode BINODE;
```

FIFO

```
typedef struct
        { BINODE *pin_val,       /* ponteiro para o ponto de inserção */
                 *pout_val;      /* ponteiro para o ponto de retirada */
        } FIFO;
```

Semáforo

```
typedef struct
        { unsigned int val;                      /* valor de contagem */
          FIFO queue;      /* fila de espera dos processos bloqueados */
        } SEMAPHORE;
```

e as variáveis globais descritas abaixo

```
static SEMAPHORE sem[200];                    /* array de semáforos */
static PCT_ENTRY pct[100];          /* tabela de controlo de processos */
static FIFO redtorun[8];   /* array das filas de espera dos processos
                                          prontos a serem executados */
static unsigned int pindex;           /* índice da entrada da PCT que
                        referencia o processo que detém o processador */
```

Finalmente, as primitivas seguintes estão também disponíveis:

Activação e inibição das interrupções

```
void interrupt_enable (void);
void interrupt_disable (void);
```

Salvaguarda e restauro do contexto do processador

```
void save_context (unsigned int pct_index);
void restore_context (unsigned int pct_index);
```

Reserva e libertação de espaço em memória dinâmica

```
void *malloc (unsigned int size);
void free (void *pnt);
```

Inserção e retirada de nós na FIFO

```
void fifo_in (FIFO *fifo, BINODE *val);
void fifo_out (FIFO *fifo, BINODE **valp);
```

Operações sobre semáforos

```
unsigned int semcreate (void);
void semdestroy (unsigned int sem_index);
void semdown (unsigned int sem_index);
void semup (unsigned int sem_index);
```

Transição de estado dos processos

```
void dispatch (void);
void prioritysuperseded (void);
void sleep (unsigned int sem_index);
void wakeup (unsigned int sem_index);
```

Scheduling do processador (selecciona o próximo processo a que o processador vais ser atribuído)

```
BINODE *sched (void);
```

1. Será que nas circunstâncias presentes há o risco de algum processo entrar em *adiamento indefinido*? Fundamente a sua resposta.
2. A figura que descreve a disciplina de *scheduling* referencia as filas de espera dos processos BLOCKED como um agregado. Explique onde é que elas estão localizadas. Fundamente a sua resposta.
3. Construa a primitiva *sched*.
4. Construa a primitiva *sleep*.