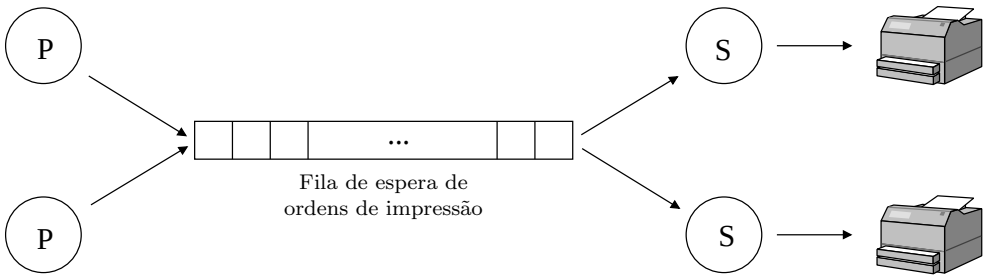
Universidade de Aveiro

Departamento de Electrónica, Telecomunicações e Informática

Sistemas de Operação

exame (parte B) (Ano Lectivo de 2008/9) 15 de Janeiro de 2009

A figura abaixo representa a organização geral de um sistema de gestão de um parque de impressoras.



Um pedido de impressão faz-se executando a primitiva

```
char* print_file(const char* filename);
```

que introduz uma ordem de impressão associada ao ficheiro `filename` na fila de espera e devolve o nome da impressora onde a impressão irá decorrer. Cada impressora activa possui um processo de sistema associado. Este processo retira sucessivamente ordens de impressão da fila e procede à impressão do conteúdo do ficheiro na respectiva impressora.

Como se pode constatar, está-se perante um modelo produtor-consumidor simples em que os processos de utilizador (P na figura), que invocam a primitiva `print_file`, são os produtores e os processos de sistema (S na figura), que executam as impressões, são os consumidores.

A interacção estabelecida supõe a existência de uma memória de tipo FIFO como meio de comunicação (a fila de espera de ordens de impressão), de dois pontos de sincronização para cada processo produtor e de um ponto de sincronização para cada processo consumidor. Os pontos de sincronização são os seguintes:

- Os processos produtores bloqueiam quando a fila de espera de ordens de impressão está cheia.
- Os processos produtores bloqueiam enquanto aguardam pela indicação da impressora onde a impressão vai ser efectuada.
- Os processos consumidores bloqueiam enquanto não há ordens de impressão.

Foram definidas as seguintes estruturas de dados.

```
/** \brief Definição de uma ordem de impressão */
typedef struct
{
    char* filename;
    int id;              /* recipiente para o id da impressora */
    int waitPoint;       /* id do semáforo de espera pelo id da impressora */
} PrintOrd;

/** \brief Definição de uma fila de espera */
#define NORDS 100
typedef struct
    PrintOrd* ord[NORDS]; /* array gerido de uma forma circular */
    int inIdx;            /* ponto de inserção */
    int outIdx;           /* ponto de retirada */
    int access;           /* id do semáforo de acesso em exclusão mútua */
    int notEmpty;         /* id do semáforo de sincronização dos consumidores */
    int notFull;          /* id do semáforo de sincronização dos produtores */
} WaitingQueue;
```

```
/** \brief Definição de uma impressora */
#define ACTIVA 0
#define INACTIVA 1
typedef struct
{
    int stat;           /* estado da impressora: ACTIVA/INACTIVA */
    char* name;         /* nome da impressora */
} Printer;

/** \brief Definição do parque de impressoras */
#define K 4
typedef struct
{
    int nap;                /* número de impressoras activas */
    Printer printers[K];    /* O id de uma impressora é o índice neste array */
    int access;             /* id do semáforo de acesso em exclusão mútua */
} PrinterPool;

/** Variáveis partilhadas */
shared PrinterPool pool;        /* parque de impressoras */
shared WaitingQueue queue;      /* Fila de espera de pedidos de impressão */
```

Estão ainda disponíveis as seguintes funções, cujos nomes são esclarecedores dos seus papeis.

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

void queue_in(WaitingQueue* queue, PrintOrd* po);
PrintOrd* queue_out(WaitingQueue* queue);

int sem_create(void);
void sem_destroy(unsigned int semid);
void sem_down(unsigned int semid);
void sem_up(unsigned int semid);
```

Responda às seguintes questões:

[2,5] 1. Nas estruturas de dados estão definidos semáforos em 5 situações. Explique a necessidades de todos estes semáforos e indique quais têm de ser criados no arranque do sistema e com que valores são inicializados.

[2,5] 2. Construa a função `queue_out` que retira uma ordem de impressão de uma fila de espera, bloqueando-se à espera se necessário.

[2,5] 3. Construa uma função com a assinatura

```
void printing_server(int id);
```

que implemente o código de um processo S. O parâmetro `id` representa a impressora. Considere que as variáveis partilhadas `pool` e `queue` estão acessíveis e que a impressora está devidamente activada. Considere ainda que dispõe de uma função com a assinatura `send_to_printer(char*)` que executa a impressão de um ficheiro.

[2,5] 4. Considere que se pretende adaptar o sistema para suportar classes de impressoras (por exemplo, impressão em frente e verso e impressão só numa face). Que alterações introduziria para que tal fosse possível. Não exclua a possibilidade de as classes não serem disjuntas: uma impressora com capacidade para impressão em frente e verso também imprime numa só face.