Universidade de Aveiro

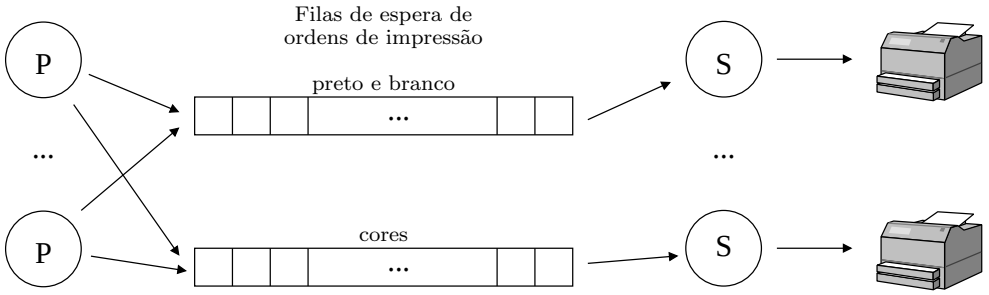
Departamento de Electrónica, Telecomunicações e Informática

Sistemas de Operação

(Ano Lectivo de 2008/9)

recurso (parte B) 29 de Janeiro de 2009

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



A impressão de um ficheiro faz-se executando a primitiva

```
int print_file(const char* filename, int class);
```

que introduz uma ordem de impressão associada ao ficheiro `filename` na fila de espera adequada e devolve a identificação (*id*) 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 correspondente à sua classe e procede à impressão do conteúdo do ficheiro na respectiva impressora. Uma impressora a cores não imprime ordens a preto e branco.

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 (as filas 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.

As ordens de impressão apenas permanecem numa fila enquanto não estão associadas a nenhum processo consumidor.

Foram definidas as seguintes estruturas de dados.

```
/** \brief Definição de uma ordem de impressão */
#define MAXNAME 255
typedef struct
{
    char filename[MAXNAME];
    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
    int ord[NORDS];         /* array de índices 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 */
} Queue;

/** Variáveis partilhadas */
shared PrintOrd po_pool[NORDS]; /* pool de ordens de impressão */
shared Queue free_queue;        /* lista de ordens de impressão livres */
shared Queue bw_queue;          /* fila de ordens de impressão a preto e branco */
shared Queue color_queue;       /* file de ordens de impressão a cores */
```

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

```
void queue_in(Queue* queue, int po_idx);
int queue_out(Queue* queue);

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

Responda às seguintes questões:

[2,5] 1. No arranque do sistema é necessário inicializar as 3 filas, `free_queue`, `bw_queue` e `color_queue`, e o reservatório (*pool*) de ordens de impressão, `po_pool`. Escreva o código da função

```
void init_free_queue();
```

que realiza a inicialização da fila `free_queue`.

[2,5] 2. Escreva o código da função `queue_in` que coloca o índice de uma ordem de impressão numa fila de espera, bloqueando-se, se necessário, à espera que o possa fazer.

[2,5] 3. Escreva o código da função

```
int print_file(const char* filename, int class);
```

Considere que o parâmetro `class` pode ter os valores `COLOR_PRINTING` e `BW_PRINTING`.

[2,5] 4. Atendendo à dimensão da *pool* de ordens de impressão e à dimensão das filas de espera, fará sentido existir o semáforo `notFull`? Justifique adequadamente a sua resposta. Proponha uma alteração do sistema que permita aproveitar a capacidade de armazenamento das filas `bw_queue` e `color_queue`.