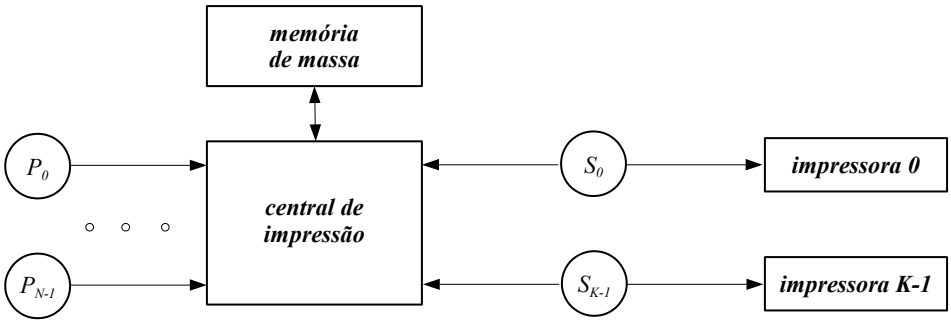
Parte A (10 valores)

1. Os sistemas de operação de uso geral actuais são tipicamente *sistemas de operação de rede*. Faça a sua caracterização.
2. Descreva o diagrama de estados do *scheduling* do processador em três níveis. Qual é o papel desempenhado por cada nível? Num sistema de tipo *batch* multiprogramado fará sentido a existência de três níveis de *scheduling*?
3. O que são monitores? Mostre como é que eles podem ser usados para resolver o problema de acesso a uma região crítica com exclusão mútua e para sincronizar processos entre si.
4. Indique as características principais de uma *organização de memória virtual*. Explique porque é que ela é vantajosa relativamente a uma *organização de memória real* no que respeita ao número de processos que correntemente coexistem e a uma melhor ocupação do espaço em memória principal.
5. Assuma que um grupo de *N* processos cooperam entre si trocando mensagens através de caixas de correio.
   a) Apresente um esquema que descreve a interacção quando N é 3.
   b) Proponha um tipo de dados `Message` para caracterização das mensagens trocadas.
   c) Proponha um tipo de dados `MailBox` para caracterização das caixas de correio.

NOTA - É do conhecimento geral que *uma imagem vale mais do que mil palavras*... Assim, sempre que seja adequado complemente as suas respostas com diagramas ou esquemas elucidativos.

Parte B (10 valores)

A figura descreve esquematicamente um sistema de gestão da impressão num parque de impressoras ligadas a um mesmo sistema computacional.



P_i, com $i = 0,1, \cdots, N-1$, representam os processos utilizador que solicitam impressões.

S_j, com $j = 0,1, \cdots, K-1$, representam os processos de sistema que transferem os dados a imprimir para a impressora que lhes está associada.

Um pedido de impressão será servido por uma qualquer das impressoras, aquela que for seleccionada internamente pela central de impressão. O conteúdo a imprimir é armazenado inicialmente em memória de massa, a impressão só começa propriamente quando o conteúdo tiver sido completamente transferido.

Os processos utilizador para usarem o serviço têm disponíveis as primitivas seguintes

```
int openPrinter(void);
```

que solicita a abertura de um canal de comunicação e que devolve os valores

>= `0` – *operação realizada com sucesso* (*o valor devolvido especifica o identificador do canal atribuído*)
`-1` – *tabela de canais de comunicação cheia* (*não se aceitam de momento mais pedidos*)

```
int writePrinter(int id, int n, char *buf);
```

que envia `n` caracteres armazenados na região de memória referenciada por `buf` para impressão no canal de comunicação `id` e que devolve os valores

`0` – *operação realizada com sucesso*
`-1` – *identificador desconhecido* (*nada é feito*)
`-2` – *canal já foi fechado* (*nada é feito*)

```
int closePrinter(int id);
```

que solicita o fecho do canal de comunicação `id`, possibilitando que a impressão propriamente dita possa vir a iniciar-se, e que devolve os valores

`0` – *operação realizada com sucesso*
`-1` – *identificador desconhecido* (*nada é feito*)
`-2` – *canal já foi fechado* (*nada é feito*) .

Os processos de sistema associados a cada impressora estão à partida bloqueados. Sempre que um processo utilizador invoca a operação `closePrinter`, um deles é acordado e vai encarregar-se do envio dos dados residentes em memória de massa para a impressora respectiva. A transferência faz-se byte a byte. Finda a tarefa, o processo liberta o canal de comunicação e volta a bloquear.

A memória de massa pode ser entendida como um *array* de T blocos, cada um permitindo o armazenamento de R bytes.

A central de impressão é definida pela estrutura de dados

```
#define  M   100
typedef struct
{ unsigned int access;      /* id do semáforo de acesso à  manipulação
                    da estrutura de dados em regime de exclusão mútua */
  unsigned int waitForWork;          /* id do semáforo de bloqueio dos
                                                  processos de sistema */
  unsigned int waitForBlock;         /* id do semáforo de bloqueio dos
            processos utilizador quando a memória de massa está cheia */
  unsigned int nComChan;        /* n.º de canais de comunicação livres */
  COM_CHAN chan[M];                /* array dos canais de comunicação */
  FIFO chanId;        /* lista ligada com as identificações dos canais
                                 que requerem processamento (impressão) */
} PRINT_CENTR;
```

em que o canal de comunicação se caracteriza por

```
#define  R   1024
typedef struct
{ int stat;                          /* indicação do estado de ocupação
                  (0 – livre / 1 – em uso (por um processo utilizador)
                                / 2 – em uso (por um processo de sistema)) */
  unsigned int nChar;                /* número de caracteres a imprimir */
  char buf[R];              /* buffer de armazenamento (correspondente
                                                  a um bloco de disco) */
  FIFO content;       /* lista ligada com as identificações dos blocos
                                    que contêm a informação a imprimir */
} COM_CHAN;  .
```

As listas ligadas indicadas acima implementam uma memória de tipo FIFO definida por

```
struct node
{ unsigned int info;                                  /* valor armazenado */
  struct node *next;                /* ponteiro de ligação (lista ligada) */
} NODE;

typedef struct
{ NODE *in;                          /* ponteiro para o ponto de inserção */
  NODE *out;                         /* ponteiro para o ponto de retirada */
} FIFO;  .
```

Como é imediato concluir do exposto, sempre que um processo utilizador sinaliza a conclusão do envio dos dados a imprimir (através da invocação da operação `closePrinter`), é necessário introduzir a identificação do canal de comunicação na lista respectiva da central de impressão. Do mesmo modo, quando não houver blocos de memória de massa livres, o processo utilizador que deles necessita, bloqueará até eles estejam de novo disponíveis.

Assuma ainda que tem à sua disposição as primitivas básicas de reserva e libertação de espaço em memória dinâmica (nível do *kernel*)

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

de manipulação de semáforos

```
int sem_create (void); ,
```

é devolvido o identificador do semáforo (o campo *val* fica a zero após a criação do semáforo)

```
void sem_destroy (unsigned int sem_id);
void sem_down (unsigned int sem_id);
void sem_up (unsigned int sem_id); ,
```

de manipulação dos FIFOs

```
void fifoInit (FIFO *f);
```

```
void fifoIn (FIFO *f, NODE *node);
NODE *fifoOut (FIFO *f);
int fifoEmpty(FIFO *f);
```

é devolvido `(1 – vazio / 0 – caso contrário)` ,

de transferência de um byte para uma impressora identificada por `id`

```
void writeP (unsigned int id, char val); ,
```

de transferência de um bloco de dados de e para a memória de massa

```
void readBlock (unsigned int blockNumber, char *buf);
void writeBlock (unsigned int blockNumber, char *buf); ,
```

e que foram definidas as variáveis seguintes

```
static PRINT_CENTR comm;                    /* central de impressão */
static FIFO freeBlocks;    /* lista ligada com as identificações dos
                           blocos da memória de massa que estão livres */  .
```

1. Considere uma situação em que existe apenas um processo utilizador a imprimir. Assuma que está a usar o canal de comunicação 0 e que foram já transferidos 2058 bytes. Quantos blocos da memória de massa foram preenchidos? Quantos bytes estão presentemente armazenados no campo `buf` de `chan[0]`? Indique o valor assumido pelos campos restantes de `chan[0]`. Justifique detalhadamente a sua resposta.
2. A *central de impressão*, tal como é descrita, impõe uma restrição ao tamanho máximo dos dados a enviar por um processo utilizador para impressão. Qual é ela? O que acontecerá se esse valor for ultrapassado? Justifique a sua resposta.
3. Construa a primitiva `openPrinter`. Comente adequadamente o código apresentado.
4. Apresente o código que descreve os processos de sistema S

   ```
   void systemPrint (unsigned int id);
   ```

   em que id é o identificador de uma impressora. Comente adequadamente o código apresentado.