ISEP – LEEC - Sistemas Computacionais 2016/2017

Ficha 7 – *Sincronização de Tarefas*

Extraia o conteúdo do ficheiro `ficha-sinc-ficheiros.zip` para o seu diretório de trabalho. Execute o comando `make` para criar os ficheiros executáveis.

1 - Analise o seguinte extrato de um programa (exame época normal 2015/2016):

```
int main() {
  int *v = mmap(NULL, sizeof(int), PROT_READ | PROT_WRITE,
   MAP_SHARED | MAP_ANONYMOUS, -1, 0);

  sem_t *psem = sem_open("/sem1", O_CREAT | O_RDWR, 0600, 1);

  v[0] = 0;
  int r, n = 0;
  while(1) {
    ns = accept(s, NULL, NULL);
    r = fork();
    ++n;
    if(r == 0) {
      r = fork();
      ++v[0];
      if(r == 0) {
        sem_wait(psem);
        sleep(3);
        sem_post(psem);
        printf("n = %d, v = %d, pid = %d, ppid = %d.\n", n, v[0], getpid(), getppid());
        return(0);
      }
      sleep(1);
      printf("%d a terminar; *v = %d.\n", getpid(), v[0]);
      exit(0);
    }
    waitpid(r, NULL, 0);
    printf("%d terminado; *v = %d, n = %d.\n", r, v[0], n);
  }
}
```

Apresente a sequência de impressões produzidas por este programa <u>após a ligação de 2 clientes</u>. Assuma que não existem interferências de outros processos no sistema, que o identificador do processo inicial é 2000 e que o(s) novo(s) processo(s) toma(m) o(s) valor(es) seguinte(s). Apresente um diagrama temporal representativo da execução do programa e justifique sucintamente. A sequência de impressões deve ser apresentada de forma destacada.

2 - Analise o programa contido em `ex2.c`. A função `myprint()` é a mesma função descrita no exercício 1 da ficha anterior.

2.1 - Utilize o mecanismo de semáforos com nome para garantir que a impressão de cada processo é enviada para o ecrã sem ser interrompida pelas impressões do outro processo.

2.2 – Ao contrário do que se observou no exercício 1 da ficha 6, basta agora fazer uma chamada à função `malloc`() (`buf=malloc(256)`) para que cada "tarefa" armazene e imprima a sua própria *string*. Porquê?

2.3 - Altere a linha

```
char *buf = malloc(8);
```

para

```
char *buf = (char *) mmap(NULL, 8, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
```

e verifique que ambos os processos passam a imprimir a mesma mensagem.

3 – Considere a API de filas de mensagem da norma POSIX (man 7 mq_overview, man mq_send, man mq_receive).

3.1 – Implemente uma biblioteca de fila de mensagens (FIFO, sem níveis de prioridade) para aplicações *multi-thread*, de acordo com as declarações apresentadas abaixo (ficheiro `mymq.h`). As funções mymq_send e mymq_receive deverão ter um comportamento análogo ao das funções mq_send e mq_receive da norma POSIX. Os bloqueios de fila cheia, no caso do mymq_send, e de fila vazia, no caso do mymq_receive, deverão ser geridos com os semáforos declarados na estrutura mymq_t, que deverá ser corretamente iniciada pela função mymq_init. O ficheiro ex3.c apresenta um exemplo de utilização das funções a implementar.

```
typedef struct
{
	int slot_size; //maximum slot size;
	int number_of_slots; //maximum slot size;
	void *slots; //malloc(slot_size*number_of_slots)
	int *data_size; //indicates the actual size of the data stored on each slot
	int oldest_slot; //index of the oldest element in the queue
	sem_t sem_msgs_in_queue; //send increments this, receive waits on this
	sem_t sem_free_slots; //receive increments this, send waits on this
	pthread_mutex_t* access;  //the accesses to the message queue
				  //must be mutually exclusive
}
mymq_t;


//returns -1 on error
int mymq_init(mymq_t *mq, int slot_size, int number_of_slots);

void mymq_send(mymq_t *mq, void *data, int size);

//returns size of received message  (in bytes)
int mymq_receive(mymq_t *mq, void *data, int size);

//releases all resources used by the queue
int mymq_destroy(mymq_t *mq);
```

3.2 – Teste a biblioteca desenvolvida com o programa do ficheiro ex3.c.