Sistemas Operativos, Exame 2, 27 de Janeiro de 2018
Soluções

IST - LEIC-A/ LEIC-T/ LETI - 2017-2018

Soluções (1/7)

Número:
Nome:

§

Programação com tarefas por troca de mensagens

Pergunta 1	

```
#define N 10
#define FILENAME_SIZE 255
#define WORD_SIZE 10

typedef struct {
  int     id;
  char    alvo[WORD_SIZE];
} argsEscrava_t;

/*-------------------------------------------------------------------
| Function: fn_escrava
-------------------------------------------------------------------*/
void *fn_escrava(void *a) {
      char codigo_pedido = 'P';
      char nome_ficheiro[FILENAME_SIZE];
      argsEscrava_t *arg   = (argsEscrava_t *) a;
      int myid = arg->id;
      char *alvo = arg->alvo;
      int   n_encontradas;


      while (1) {
          /* envia mensagem a pedir nome do ficheiro */
          enviarMensagem ( myid, 0, codigo_pedido, sizeof(codigo_pedido) );

          /* recebe nome do ficheiro */
          receberMensagemDeQualquerOrigem ( myid, NULL, nome_ficheiro, FILENAME_SIZE);

           /* processa o ficheiro */
           n_encontradas =  conta_palavra (nome_ficheiro, alvo);

           /* envia para o acumulador */
           enviarMensagem( myid, N+1,   &n_encontradas, sizeof(int));
      }
      return 0;
}

/*-------------------------------------------------------------------
| Function: fn_acumuladora (já completa)
-------------------------------------------------------------------*/
void *fn_acumuladora(void *a) {
      int  n_encontradas, origem, total_acumulado = 0;
      while (1) {
          receberMensagemDeQualquerOrigem (N+1, &origem, &n_encontradas, sizeof(int));
          total_acumulado = total_acumulado + n_encontradas;
          printf ("Recebeu de %d; Total acumulado=%d\n", origem, total_acumulado);
      }
      return NULL;
}
```

Soluções (2/7)

§

```
/*-------------------------------------------------------------------
| Function: main
-------------------------------------------------------------------*/
int main (int argc, char** argv) {
    char*           nome_ficheiro;
    argsEscrava_t   escrava_args[N];
    pthread_t       escravas[N];
    pthread_t       acumuladora;
    char *palavra_alvo = argv[1]; //testes aos argumentos omitidos
    int origem;
    char codigo_msg;

    /* Inicializa biblioteca de troca de mensagens  (capacidade do canal, numero de tarefas comunicantes) */
    inicializarMPlib(CHANNEL_SZ, N+2);

    /* cria as tarefas escravas */
    for (t=1; t <= N; t++) {
        escrava_args[t-1].id = t;
        strcpy(escrava_args[t-1].alvo, palavra_alvo);
        pthread_create(&escravas[t-1], NULL, fn_escrava, escrava_args[t-1]);
    }

    /* cria a tarefa acumuladora */
    pthread_create(&acumuladora, NULL, fn_acumuladora, NULL );

    while (1) {
         nome_ficheiro = proximo_ficheiro ();

        /* recebe pedido de uma tarefa escrava */
        receberMensagemDeQualquerOrigem (0, &origem, &codigo_msg, sizeof(codigo_msg);

        if (codigo_msg == 'P') {
             /* envia nome do ficheiro */
             enviaMensagem (0, origem, nome_ficheiro, strlen(nome_ficheiro)+1);
        }
    }
}
```

§

Programação com tarefas por memória partilhada

Pergunta 2

```
//Declaração/inicialização de variáveis globais}
sem_t s;
sem_init(&s, .., max_A);

iniciaAcessoCatA() {
    sem_wait(&s);
}

terminaAcessoCatA() {
    sem_post(&s);
}
```

Soluções (3/7)

§

Programação com tarefas por memória partilhada (cont.)

Pergunta 3

```
//Declaração/inicialização de variáveis globais}
pthread_mutex_t m; pthread_cond_t c;
pthread_mutex_init(&m, NULL);
pthread_cond_init(&c, NULL);
int c_A = 0;

iniciaAcessoCatA() {
    pthread_mutex_lock(&m);
    while (c == max_A) pthread_cond_wait(&c, &m);
    c_A ++;
    pthread_mutex_unlock(&m);
}


terminaAcessoCatA() {
    pthread_mutex_lock(&m);
    c_A --;
    pthread_cond_signal(&c);
    pthread_mutex_unlock(&m);
}
```

Pergunta 4	Diferentes respostas possíveis. Por exemplo: Obter ambos os trincos no início de ambas as funções, pela mesma ordem: `fechar(m_A); fechar(m_B);` (em ambas as funções).

§

Programação com processos

Pergunta 5	Programa A ☐ Programa B ☑ Ambos ☐ Nenhum ☐
Pergunta 6	Programa A ☐ Programa B ☑ Ambos ☐ Nenhum ☐

Pergunta 7

```
void trataSignal(int s) {
    printf("acknowledged\n");
    exit(0);
}


int main() {
    int pid[N], i;

    for (i=0; i<N; i++) {
        pid[i] = fork();
        if (pid[i] == 0) {
            signal(SIGUSR1, trataSignal);
            printf("f(\%d) iniciou\n", i);
            f(i);
            printf("f(\%d) terminou\n", i);
            exit(0);
        }
    }

    wait(NULL);
    for (i=0; i<N; i++)
        kill(SIGUSR1, pid[i]);

    for (i=0; i<N-1; i++)
        wait(NULL);
    exit(0);
}
```

§

Gestor de Processos

Pergunta 8	Cada processo reduz a sua prioridade de forma inversamente proporcional à ordem de criação de cada processo. Consequentemente, os processos terminarão pela ordem inversa pela qual foram criados.

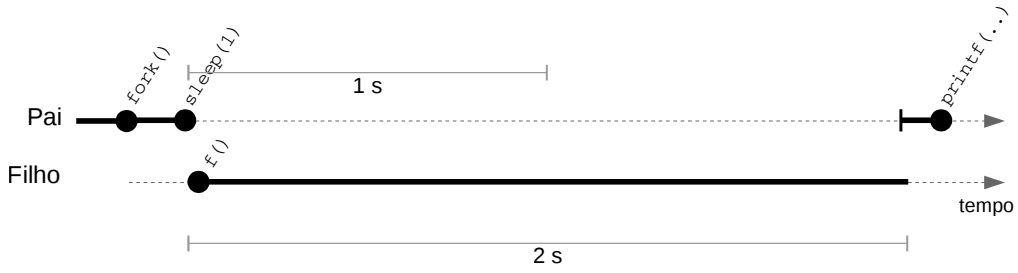
Soluções (5/7)

§

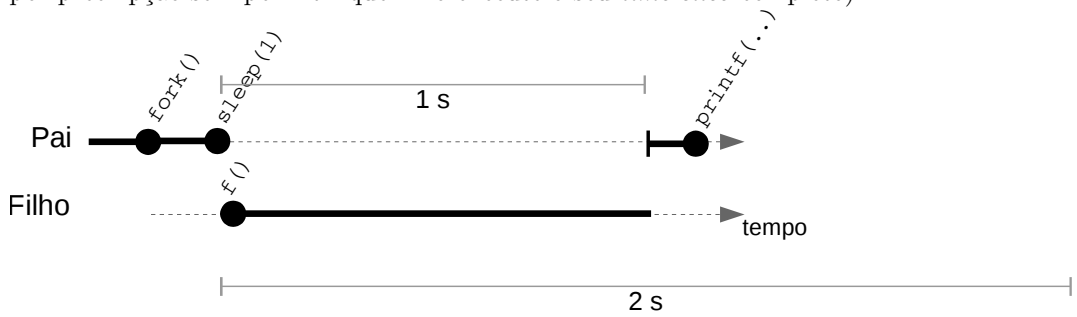
Gestor de Processos

Pergunta 9

Exemplo sem preempção (processo filho ganha o CPU e só o liberta ao fim de 2s):



Exemplo com preempção (processo pai desbloqueia-se após 1s e recebe o CPU por preempção sem permitir que filho execute o seu *time-slice* completo):



Nota: em ambos os exemplos, só se apresenta a execução até ao momento em que o pai chama a função *printf*.

§

Gestão de Memória

Pergunta 10	2^{16} páginas

Pergunta 11

Acesso	Endereço Virtual	FP	Endereço Real	Bit Presença	Acedido (R)	Dirty (M)	Excepção lançada
Leitura	`0x0001A345`	N	`0x0011A345`	1	1	0	
Leitura	`0x0002AGFC`	S	`0x0AAAAGFC`	1	1	0	Falta de Página
Leitura	`0x0002AABD`	N	`0x0AAAAABD`	1	1	0	
Escrita	`0x0003AA4F`	N	`0x0033AA4F`	1	1	1	
Leitura	`0x00041251`	N	`0x00311251`	1	1	1	
Escrita	`0x00000000`	N		0	0	0	Violação de acesso

Pergunta 12	Sim, quando periodicamente o processo paginador percorre todas a tabela de páginas e re-coloca o bit a 0.
Pergunta 13	Sim, assim que o conteúdo página for salvaguardado em disco.

§

Sistema de Ficheiros

Pergunta 14

Inode	Tamanho Entrada	Tamanho do Nome	Tipo	Nome
2	12	1	2	.\0\0\0
2	12	2	2	.. \0\0
11234	12	3	2	tmp\0
11111	16	5	1	a.txt\0\0\0

Inode	Tamanho Entrada	Tamanho do Nome	Tipo	Nome
11234	12	1	2	.\0\0\0
2	12	2	2	.. \0\0
22222	16	5	1	b.txt\0\0\0
33333	16	5	1	c.txt\0\0\0

§

Pergunta 15

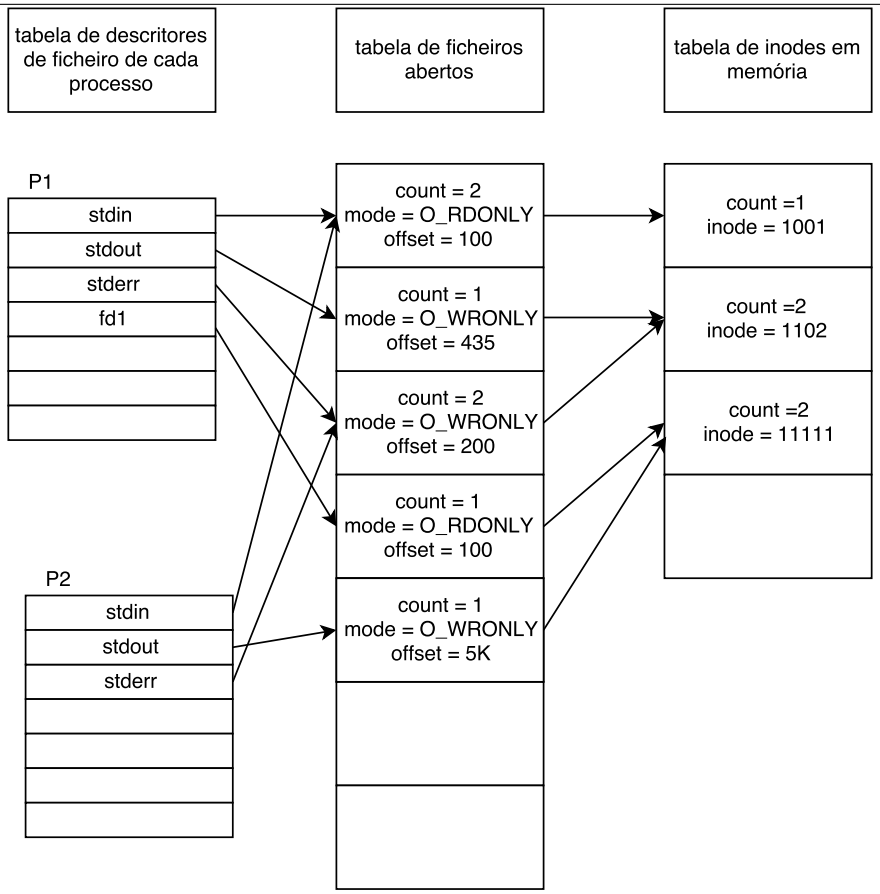
Inode	Tamanho Entrada	Tamanho do Nome	Tipo	Nome
2	12	1	2	.\0\0\0
2	12	2	2	.. \0\0
11234	12	3	2	tmp\0
11111	16	5	1	a.txt\0\0\0

Inode	Tamanho Entrada	Tamanho do Nome	Tipo	Nome
11234	12	1	2	.\0\0\0
2	12	2	2	.. \0\0
22222	16	5	1	b.txt\0\0\0
11111	16	5	1	d.txt\0\0\0

Soluções (7/7)

§

Pergunta 16	



Pergunta 17	Como são escritos 2048 bytes adicionais no final do ficheiro, este passa a ter 5Kbytes de tamanho; ou seja, passa a ocupar 2 blocos de 4Kbytes (em vez de 1). Consequentemente, o inode passa a ter a segunda entrada do bloco de índices a referenciar o bloco 500. Adicionalmente, o tamanho do ficheiro e a data do último acesso são também alterados no inode.