

Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática

Sistemas Operativos

Ano Letivo 2020/21

Trabalhos Laboratoriais de Programação

Elaborado em: 26/06/2021

Daniel Santos | Nº2019133865

Rui Marques | Nº2019134520

Índice

List of Figures.....	ii
1 Introdução.....	1
2 Threads.....	2
3 Sincronização.....	8
4 IPC - Message Queues.....	11
5 Conclusão.....	14
5.1 Strength.....	14
5.2 Weaknesses.....	14
5.3 Opportunities.....	14
5.4 Threats.....	14
5.5	14
6 Referências.....	15

List of Figures

FIGURA 1 - MENU THREADS.....	2
FIGURA 2 - FUNÇÃO TRANSFORMALINHASCOLUNAS.....	3
FIGURA 3 - FUNÇÃO FICHA4THREAD2.....	4
FIGURA 4 - COMPILAÇÃO E EXECUÇÃO DA FICHASSO.C.....	5
FIGURA 5 - TRANSFORMACAO DE COLUNAS EM LINHAS.....	5
FIGURA 6 - PL_PROGRAMACAO_00_DATABASE_INV.CSV.....	6
FIGURA 7 - EXTRACAO E GRAVACAO DE LINHAS.....	6
FIGURA 8 - PL_PROGRAMACAO_04_THREADS.CSV.....	7
FIGURA 9 - PTHREAD_MUTEX_INIT.....	8
FIGURA 10 - PTHREAD_MUTEX_DESTROY.....	8
FIGURA 11 - PTHREAD_MUTEX_LOCK.....	8
FIGURA 12 - PTHREAD_MUTEX_UNLOCK.....	8
FIGURA 13 - SEM_INIT.....	8
FIGURA 14 - SEM_DESTROY.....	9
FIGURA 15 - SEM_WAIT.....	9
FIGURA 16 - SEM_POST.....	9
FIGURA 17 - COMPILAÇÃO DO CODIGO.....	9
FIGURA 18 - MENUS DA FICHA 5.....	9
FIGURA 19 - PEDIDOS DE INFORMAÇÃO.....	10
FIGURA 20 - RESULTADO FINAL.....	10
FIGURA 21 - STRUCT.....	11
FIGURA 22 - CODIGO FICHA 6.....	12
FIGURA 23 - COMPILAÇÃO DO FICHEIRO.....	12
FIGURA 24 - EXECUÇÃO DO PROGRAMA DA FICHA 6.....	13

1 Introdução

No âmbito da disciplina de Sistemas Operativos foi pedido para fazer a realização de 3 fichas abordando vários conceitos, como a criação de threads, a sua sincronização através de *mutex* e *inter-process communication* (IPC). Vais ser apresentado a resolução dos vários exercícios, sempre apresentando as respetivas capturas de ecrã para comprovar os vários resultados obtidos.

Tomámos a iniciativa de explicar primeiro todos os conceitos da ficha e depois apresentar o código com os seus comentários devidos.

O trabalho foi realizado no Fedora usando uma máquina virtual.

2 Threads

```
void Threads(){
    int opcao;
    pthread_t thread1,thread2,thread3;
    int c1=0, c2=0;

    printf("1.Transformacao de colunas em linhas  2.Extracao e gravacao de linhas\n");
    scanf("%d",&opcao);
    switch (opcao) {
        case 1:
            pthread_create(&thread1, NULL, TransformaLinhasColunas, NULL);
            pthread_join(thread1, NULL);
            break;

        case 2:
            system(" > PL_Programacao_04_Threads.csv ");
            printf("\nGeração do ficheiro PL_Programacao_04_Threads.csv efetuada com sucesso!");

            printf("\nInsira o nome do ficheiro: ");
            scanf("%s", ficheiro);
            printf("\nInsira a primeira coluna: ");
            scanf("%d", &c1);
            printf("\nInsira a segunda coluna: ");
            scanf("%d", &c2);
            printf("\nInsira o número de repetições: ");
            scanf("%d", &repeticoes);

            pthread_create(&thread2, NULL, ficha4thread2, &c1);
            pthread_join(thread2, NULL);

            pthread_create(&thread3, NULL, ficha4thread3, &c2);
            pthread_join(thread3, NULL);
            break;
    }
}
```

Figura 1 - Menu threads

Começámos primeiro por fazer o menu das threads:

Criámos duas opções transformação de colunas para linhas e extração e gravação de linhas.

Se escolhermos a primeira opção iremos por uma thread a utilizar a função TransformaLinhasColunas():

```
void *TransformaLinhasColunas(){
    char auxFicheiro[500];
    char novoFicheiro[500];
    char auxToken[500][500];
    char comando[500];
    int i=0;
    printf("Insira o nome do ficheiro:");
    fflush(stdin);
    scanf("%s",&ficheiro);

    strcat(auxFicheiro, ficheiro);
    char * token = strtok(ficheiro, ".");

    do{
        strcat(auxToken[i], token);
        token = strtok(NULL, ".");
        i++;
    }while(token !=NULL);

    strcat(novoFicheiro, auxToken[i-2]);
    strcat(novoFicheiro, "_inv.");
    strcat(novoFicheiro, auxToken[i-1]);

    strcat(comando, "awk -F, '{for(i=1; i<=NF; i++){A[NR, i]=$i}; if(NF>n){n=NF";
    strcat(comando, auxFicheiro);
    strcat(comando, " > ");
    strcat(comando, novoFicheiro);
    system(comando);

    printf("Geracao do ficheiro com troca de colunas com linhas efetuada com su
    char a;
    scanf("%s",&a);
    pthread_exit(0);
```

Figura 2 - Função TransformaLinhasColunas

Para mudar o nome do ficheiro utilizou-se tokens pois facilita o processo de acrescentar “_inv” ao ficheiro.

Para mudar o conteúdo o ficheiro contém usámos o awk.

Se escolhermos a segunda opção iremos por duas threads a utilizar funções diferentes. Ambas têm um objetivo parecido à primeira função, só que desta vez, escolhe-se um coluna para cada thread escrever no ficheiro PL_Programacao_04_Threads.csv quantas vezes o utilizador optar:

```
void *ficha4thread2(void *coluna1){
    char comando[500];
    char buffer1[500];
    char buffer2[500];

    int c1 = *(int*)coluna1;

    strcat(comando, "awk -F ' ' '{linha[NR]=$}");
    sprintf(buffer1, "%d", c1);
    strcat(comando, buffer1);
    strcat(comando, "} END{for(x=0; x<");
    sprintf(buffer2, "%d", repeticoes);
    strcat(comando, buffer2);
    strcat(comando, "; x++){for(i=0; i<NR; i++) {printf \"%s \", linha[i]}{printf \"\\n\\n\"}} }' ");
    strcat(comando, ficheiro);
    strcat(comando, " >> PL_Programacao_04_Threads.csv");
    system(comando);
    printf("\nThread2 %d do processo %d escreveu com sucesso no ficheiro PL_Programacao_04_Threads.csv, a coluna %d repetida %d vezes!\n", getpid(), getpid(), c1, repeticoes);
    char a;
    scanf("%s", &a);
    pthread_exit(NULL);
}
```

Figura 3 - Função ficha4thread2

Aqui podemos observar que a thread2 está a receber como parâmetro a primeira coluna para copiar para o ficheiro.

Usámos novamente o comando awk para copiar de um ficheiro para outro e repetir quantas vezes nos apetecer.

A segunda thread roda uma função diferente mas de igual funcionalidade sendo que desta vez irá pegar na segunda coluna.

```
[root@fedora ProgramasC]# gcc fichasS0.c -o fichasS0 -lpthread
[root@fedora ProgramasC]# ./fichasS0
Bem vindo!
1.Gestao de processos  2.Criacao de processos  3.Signals  4.Threads  0.Sair
█
```

Figura 4 - Compilação e execução da fichasS0.c

Chegando agora à parte prática iremos primeiro demonstrar como compilar e executar ficheiros que usam threads:

Depois escolhemos a opção Threads, Transformacao de colunas em linhas e introduzimos o ficheiro que desejamos ler e inverter para um novo ficheiro:

```
Bem vindo!
1.Gestao de processos  2.Criacao de processos  3.Signals  4.Threads  0.Sair
4
1.Transformacao de colunas em linhas  2.Extracao e gravacao de linhas
1
Insira o nome do ficheiro:PL_Programacao_00_Database.csv
Geracao do ficheiro com troca de colunas com linhas efetuada com sucesso. Insira
qualquer tecla para continuar...
0
1.Gestao de processos  2.Criacao de processos  3.Signals  4.Threads  0.Sair
```

Figura 5 - Transformacao de colunas em linhas

Podemos observar que o ficheiro inicial foi invertido para um novo ficheiro:

fichasSO.c	PL_Programacao_00_Database_inv.csv
age, 55, 56, 28, 24, 25, 45, 42, 35, 46, 34, 55, 28, 31, 42, 35, 52, 21	
39, 39, 40, 25, 29, 23, 48, 24, 59, 31, 70, 37, 59, 38, 56, 48, 28, 20,	
52, 55, 58, 61, 64, 45, 66, 31, 31, 36, 20, 40, 38, 51, 47, 31, 34, 35,	
38, 31, 37, 58, 37, 73, 44, 66, 50, 30, 41, 57, 71, 53, 38, 46, 61, 48,	
28, 25, 31, 46, 34, 26, 42, 58, 55, 63, 59, 43, 41, 34, 19, 27, 38, 59,	
52, 56, 48, 38, 34, 46, 60, 31, 30, 41, 49, 27, 25, 28, 30, 26, 32, 38,	
29, 47, 48, 54, 42, 38, 52, 56, 19, 65, 29, 30, 41, 52, 59, 29, 51, 42,	
35, 61, 56, 54, 47, 63, 39, 51, 32, 45, 36, 62, 64, 42, 31, 24, 38, 45,	

Figura 6 - PL_Programacao_00_Database_inv.csv

Depois fomos para a segunda opção (Extração e gravação de linhas) onde introduzimos o nome do ficheiro a ler a coluna 1 e 2 e o número de repetições tendo como feedback as informações da thread:

```
1.Gestao de processos  2.Criacao de processos  3.Signals  4.Threads  0.Sair
4
1.Transformacao de colunas em linhas  2.Extracao e gravacao de linhas
2
Geração do ficheiro PL_Programacao_04_Threads.csv efetuada com sucesso!
Insira o nome do ficheiro: PL_Programacao_Database.csv

Insira a primeira coluna: 2

Insira a segunda coluna: 4

Insira o número de repetições: 10
sh: linha 1: PL_Programacao_00_Database.csvawk: comando não encontrado

Thread2 185009728 do processo 8713 escreveu com sucesso no ficheiro PL_Programac
ao_04_Threads.csv, a coluna 2 repetida 10 vezes! Insira qualquer tecla para cont
inuar...
```

Figura 7 - Extração e gravação de linhas

Figura 8 -

```
marital 1 0 1 1 0 1 0 0 0 1 1 0 1 0
1 0 1 1 0 1 0 0 1 0 0 0 1 0 1 1 0 0
0 0 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1
1 1 1 0 1 0 0 0 1 1 1 0 0 0 1 0 1 1
0 1 1 1 0 1 0 1 1 0 0 0 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 0 1 0
1 1 1 0 0 1 0 1 0 1 1 1 1 0 1 0 1 0
1 0 1 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1
0 0 1 0 0 1 0 1 0 1 0 0 0 1 1 0 0 0
1 1 0 0 1 1 1 0 0 1 0 1 0 1 0 1 1 0
0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 1 0
1 0 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0
0 0 1 0 0 0 0 1 0 1 1 1 0 1 0 0 0 1
1 0 1 0 1 0 1 0 1 1 0 0 1 0 0 1 0 0
1 0 1 1 1 1 0 0 1 1 1 0 1 1 0 0 0 0
1 0 0 1 1 1 1 0 0 0 1 0 1 1 1 1 0 1
0 1 1 1 1 0 1 0 0 1 0 1 1 1 0 0 1 1
0 0 0 1 1 0 0 0 0 1 1 0 1 1 1 1 0 0
1 0 1 0 1 0 0 1 1 1 1 1 1 1 0 0 0 1
1 1 1 1 1 0 1 0 1 1 1 1 0 0 1 1 0 0
1 0 1 1 0 0 1 0 1 1 1 1 1 0 0 1 1 1
1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0 1 0
1 0 1 1 0 1 1 1 1 0 0 0 1 0 1 1 1 0
0 1 1 1 0 1 0 0 1 1 1 0 0 0 1 0 1 0
0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 1
1 0 1 1 0 1 0 0 1 0 1 1 1 0 1 0 0 0
marital 1 0 1 1 0 1 0 0 0 1 1 0 1 0
```

PL_Programacao_04_Threads.csv

Podemos observar que a palavra “marital” repete (repetiu 10 vezes), então está a funcionar bem.

3 Sincronização

Esta ficha é muito parecida com a segunda parte da ficha 4, tendo sido introduzido os mutex e os semáforos.

Na primeira parte foi adicionado no menu as funções `pthread_mutex_init` e `pthread_mutex_destroy`, sendo a primeira para iniciar o mutex e a segunda para a destruir, e na função para a thread as funções `pthread_mutex_lock`, para travar a função para o uso por parte de apenas uma thread e a função `pthread_mutex_unlock`, para destravar a mesma.

```
pthread_mutex_init(&mutex, NULL);
```

Figura 9 - pthread_mutex_init

```
pthread_mutex_destroy(&mutex);
```

Figura 10 - pthread_mutex_destroy

```
pthread_mutex_lock(&mutex);
```

Figura 11 - pthread_mutex_lock

```
pthread_mutex_unlock(&mutex);
```

Figura 12 - pthread_mutex_unlock

Na segunda parte foi adicionado no menu as funções `sem_init` e `sem_destroy`, sendo a primeira para iniciar o semáforo e a segunda para o destruir, e na função para a thread as funções `sem_wait`, para forçar as threads que querem utilizar a função enquanto outra a está a usar, a esperar e a função `sem_post`, para permitir a outra thread usar a função.

```
sem_init(&sem, 0, 1);
```

Figura 13 - sem_init

```
sem_destroy(&sem);
```

Figura 14 - sem_destroy

```
sem_wait(&sem);
```

Figura 15 - sem_wait

```
sem_post(&sem);
```

Figura 16 - sem_post

Já na pratica primeiro compilamos o ficheiro e executamo-lo.

```
rui@RuiM:~$ gcc -pthread -o fichasS0 fichasS0.c
```

Figura 17 - Compilação do Código

Depois escolhemos a opção Sincronização e temos as duas partes da ficha.

```
rui@RuiM:~$ ./fichasS0
Bem vindo!
1.Gestao de processos  2.Criacao de processos  3.Signals  4.Threads  5.Sincroniz
acao  6.IPC  0.Sair
5
1.Extracao e gravacao de colunas(pthread_mutex)  2.Extracao e gravacao de colun
as(sem)
█
```

Figura 18 - Menus da ficha 5

Independetemente da opção que escolhemos é pedido o nome do ficheiro base, as duas colunas e o numero de repetições.

```
1.Extracao e gravacao de colunas(pthread_mutex) 2.Extracao e gravacao de colunas(sem)
1
Geração do ficheiro PL_Programacao_05_Threads.csv efetuada com sucesso!
Insira o nome do ficheiro: PL_Programacao_00_Database.csv

Insira a primeira coluna: 1

Insira a segunda coluna: 2

Insira o número de repetições: 3
2
Thread -1511315712 do processo 2418916 escreveu com sucesso no ficheiro PL_Programacao_05_Threads.csv, a coluna 2 repetida 3 vezes! Insira qualquer tecla para continuar...1
k
Thread -1502923008 do processo 2418916 escreveu com sucesso no ficheiro PL_Programacao_05_Threads.csv, a coluna 1 repetida 3 vezes! Insira qualquer tecla para continuar...k
```

Figura 19 - Pedidos de Informação

Também independentemente da opção escolhida o resultado vai ser o mesmo.

[illegible]

Figura 20 - Resultado Final

4 IPC - Message Queues

Começamos por criar uma estrutura contendo o uma variavel para o tipo da mensagem, e outra para o texto da mensagem.

```
struct mensagem{  
    long tipo;  
    char texto[100];  
}mensagem;
```

Figura 21 - Struct

Depois criamos a queue de mensagens e criamos um processo filho, o processo pai lê as mensagens introduzidas pelo utilizador e manda-as para a queue de mensagens, o processo filho abre o ficheiro para a escrita e vai escrevendo as mensagens que recebe da queue de mensagens, no fim fecha o ficheiro e escreve as mensagens que foram introduzidas no ficheiro, no ecrã.

```
void ficha6(){
    int msgid;
    msgid = msgget((key_t) 1234, 0666 | IPC_CREAT);
    if(fork() == 0)
    {
        FILE *fp;
        fp = fopen("PL_Programacao_06_IPC.dat", "wb");
        do {
            msgrcv(msgid, &mensagem, sizeof(mensagem.texto), 1, 0);
            fprintf(fp, "%s\n", mensagem.texto);
        } while(strcmp(mensagem.texto, "terminar") && strcmp(mensagem.texto, "Terminar"));

        msgctl(msgid, IPC_RMID, 0);
        fclose(fp);
        char conteudo[100];
        fp = fopen("PL_Programacao_06_IPC.dat", "rb");
        while (fscanf(fp, "%s\n", conteudo) != EOF) {
            printf("%s\n", conteudo);
        }
        exit(0);
    }else
    {
        mensagem.tipo = 1;
        do {
            printf("Write Data : ");
            scanf("%s\n", &mensagem.texto);
            msgsnd(msgid, &mensagem, sizeof(mensagem.texto), 0);
        } while(strcmp(mensagem.texto, "terminar") && strcmp(mensagem.texto, "Terminar"));

        wait(NULL);
    }
}
```

Figura 22 - Código Ficha 6

Na prática primeiro compilamos o ficheiro e executamo-lo.

```
rui@RuiM:~$ gcc -pthread -o fichasSO fichasSO.c
```

Figura 23 - Compilação do ficheiro

Depois escolhemos a opção Envio de mensagens entre processos, e começamos a escrever as mensagens que queremos enviar, para parar de enviar as mensagens enviamos a mensagem “terminar” ou “Terminar”, por fim o processo filho vai escrever as mensagens enviadas no ecrã. Para sair do sub-menu é só escolher a opção Sair.



```
rui@RuiM:~$ ./fichasSO
Bem vindo!
1.Gestao de processos  2.Criacao de processos  3.Signals  4.Threads  5.Sincroniz
acao  6.IPC  0.Sair
6
1.Envio de mensagens entre processos  2. Sair
1
Write Data : terminar
a
Mensagens recebidas:
terminar
1.Envio de mensagens entre processos  2. Sair
█
```

Figura 24 - Execução do programa da Ficha 6

5 Conclusão

Damos assim por concluído a realização das últimas 3 fichas de programação de SO, sendo que conseguimos fazer tudo e não temos críticas externas a fazer devido a um ótimo enunciado.

5.1 Strength

Ao longo da realização das várias fichas foi sempre feito um estudo inicial daquilo que estávamos a abordar, o que facilitou bastante depois ao fazer os vários exercícios e ajudou a solidificar o que aprendemos ao longo das aulas.

5.2 Weaknesses

Desta vez adaptámo-nos melhor ao desenvolvimento e compreensão do trabalho, mas tivemos um pouco de dificuldades com o comando awk.

5.3 Opportunities

O professor poderia disponibilizar um pedaço da aula (15/30 minutos) a explicar certos pormenores no awk.

5.4 Threats

Sem nada a indicar.

6 Referências

- PL - Programacao - 04 - Threads
- PL - Programacao - 05 - Sincronização
- PL - Programacao - 06 - IPC - Message Queues
- Beginning Linux Programming 4th edition - Wrox.pdf
- <https://stackoverflow.com/questions/1352749/multiple-arguments-to-function-called-by-pthread-create>
- <https://stackoverflow.com/questions/44645564/awk-fatal-cannot-open-file-for-reading-no-such-file-or-directory-in-for-loop>
- <https://www.inf.pucrs.br/~pinho/Laprol/Fflush/CorrigeScanfGets.html>