Universidade Fernando Pessoa

Sistemas Operativos

Trabalho Prático

2019/2020

Patricia Rodrigues

35365

Índice

Objetivos

- Aplicar os conhecimentos aprendidos nas aulas de Sistemas Operativos.
- Esta etapa do trabalho implica programar o “myfind”. O programa “myfind” deve aceitar os argumentos anteriormente indicados e efetuar a pesquisa de todas as ocorrências. A pesquisa deve ser feita com recurso a tarefas.

Fase 2:Ponto a.

(5%) Devem fazer o parse dos argumentos.

- Todos os argumentos ficam guardados dentro de uma estrutura. São separados por onde começa a procura, opções e o que procura.

```
FIND parse(FIND find,int argc, const char * argv[]){

    //find [onde começar] [ [opções] [o que procurar] ]
    int i,j;
    strcpy(find.ondeComeca, argv[2]);
    for(i=3,j=0;i<argc-1;i++,j++){
        strcpy(find.op[j], argv[i]);
    }
    strcpy(find.procura, argv[i]);
    return find;
}
```

Fase 2:Ponto b.

(30%) Devem ser criadas “ n threads ”, cada tarefa deve consumir um diretório. Ao encontrar um novo directório a “thread” deve criar uma nova tarefa para consumir esse novo directório.

- Neste ponto é criada n threads e sempre que se encontra um novo diretório cria se uma nova thread para esse diretório imprimindo as entradas desse diretório.

```c
void* thread_func (void* t_data)
{
  /* Cast the cookie pointer to the right type.  */
    T_DATA* t = (T_DATA*) t_data;

    printf("ThreadID = %d; arg[]= %s\n",t->threadID,t->arg);
    DIR * dir;
    struct dirent * entry;
    char aux[100]="/Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/";
    strcat(aux, t->arg);
    printf("\n\n%s\n\n",aux);
    if((dir=opendir(aux))==NULL)
        perror("opendir() error");
    else{
        T_DATA thread1_args;
        thread1_args.threadID = t->threadID+1;
        strcpy(thread1_args.arg,t->arg);
        pthread_t thread1_id;
        puts("contents of root:");
        while ((entry = readdir(dir))!=NULL) {
            printf("\t\t%s\n",entry->d_name);
            if(strcmp(entry->d_name, ".")!=0 && entry->d_name[0]!='.'){
                strcpy(thread1_args.arg,entry->d_name);
                printf("Create thread for: %s",entry->d_name);
                pthread_create (&thread1_id, NULL, &thread_func, &thread1_args);
                pthread_join (thread1_id, NULL);
            }
        }
        closedir(dir);
    }
  return NULL;
}
```

```
Patricias-MBP:MyFind faculdade$ gcc main.c -o main
Patricias-MBP:MyFind faculdade$ ./main find WS1 -name ws2
ThreadID = 1; arg[]= WS1


/Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS1

contents of root:
                .
                ..
                w1ex1
Create thread for: w1ex1
                .DS_Store
                w1ex2
Create thread for: w1ex2
ThreadID = 2; arg[]= w1ex2


/Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/w1ex2

ThreadID = 2; arg[]= w1ex2


/Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/w1ex2

opendir() error: No such file or directory
opendir() error: No such file or directory
Patricias-MBP:MyFind faculdade$
```

Fase 2:Ponto c.

(45%) Devem ser criadas “n threads consumidoras”, e “1 thread produtoras”. A tarefa produtora deve produzir diretórios para serem consumidos pelas threads consumidoras. Quando uma tarefa consumidora acaba a procura no diretório corrente deve consultar se existe mais diretórios para consumir. Quando todas as tarefas concluírem a procura, a “main thread” deve saber quantas correspondências cada tarefa satisfez.

```c
//int main(int argc, const char * argv[]){
   int main(){
        const char * argv[]={"main.c","find","WS2","-iname","daTa.txt"};
        int argc = 5;
       FIND find = {.ondeComeca ="", .op = {""}, .procura=""};
       find = parse(find, argc, argv);
       initBuffer();
       /* 1 produtor */
       T_DATA t ={.arg=""};
       strcpy(t.arg, path);
       strcat(t.arg, find.ondeComeca);
       printf("Criei o thread produtor\n");
       pthread_create (&t.threadID, NULL, &produtor, &t);

       pthread_join(t.threadID, NULL);

       /*int i;
       for(i=0;i<10;i++){
           printf("Buf[%d]=%s\n",i,buf[i]);
       }*/

       /* N consumidores*/
       int i;
       T_DATA id;
       pthread_t consID[N];
       for(i=0;i<N;i++){
           printf("Criei o thread consumidor[%d]\n",i);
           pthread_create (&id.threadID, NULL, &consumidor, (void*)&find);
           consID[i]=id.threadID;
       }
       for(i=0;i<N;i++){
           pthread_join(consID[i], NULL);
           printf("Recebi %d correspondencias do consumidor[%d]\n",correspondencias[i][0],i);
       }
    return 0;
}
```

```c
/**
 Função do produtor onde vai descobrir diretórios
 */
void * produtor(void * param)
{
    T_DATA* t = (T_DATA*)param;
    DIR * dir;
    struct dirent * entry;
    char aux[TAM] ="";
    strcpy(aux, t->arg);
    if(strcmp(t->arg, ".")!=0){
        strtok(aux, ".");
    }
    //printf("\n\n%s\n\n",aux);
    if((dir=opendir(aux))==NULL){
            perror("opendir() error");
            return NULL;
    }
    // produz o diretório corrente
    produz(aux);
        while ((entry = readdir(dir))!=NULL) {
            //printf("\t\t%s\n",entry->d_name);
            if(strcmp(entry->d_name, ".")!=0 && entry->d_name[0]!='.' && entry->d_type==4){
                strcat(aux, "/");
                strcat(aux,entry->d_name);
                // produz os diretórios dentro do diretório corrente
                T_DATA taux ={.threadID=t->threadID,.arg=""};
                strcpy(taux.arg, aux);
                produtor((void*)(&taux));
            }
        }
        closedir(dir);
    pthread_exit(param);
}
```

```c
/**
    Função que mete na matriz os diretórios para serem consumidos
 */
void produz(char * dir) {
    DIR * d;
    if((d=opendir(dir))==NULL){
        perror("opendir() error");
        return;
    }
        int i;
    char aux[100]="";
        for(i=0;i<TAM;i++){
            strcpy(aux, buf[i]);

            if(strcmp(aux,"0")==0){
                strcpy(buf[i], dir);
                return;
            }
        }

}
```

```c
void * consumidor(void * param)
{
    int i=0,consId=consumidorID++;
    char aux[100]="";
    FIND * find=(FIND*)param;
    while(1) {
        semaphore_wait(semPodeCons);
        pthread_mutex_lock(&trinco_c);
        // se tiver vazio termina o thread senao retorna o indice do diretorio
        if( (i=isEmpty())==-1){
            pthread_mutex_unlock(&trinco_c);
            semaphore_signal(semPodeProd);
            pthread_exit(NULL);
        }
        // consome um diretório

               strcpy(aux, buf[i]);
               if(strcmp(aux,"0")!=0){
                   printf("Consumidor[%d] consumiu: %s\n",consId,aux);
                   strcpy(buf[i], "0");
               }
        pthread_mutex_unlock(&trinco_c);
        semaphore_signal(semPodeProd);
        DIR *dir;
        struct dirent * entry;
        if((dir=opendir(aux))==NULL){
            perror("opendir() error:");
            break;
        }
        while ((entry = readdir(dir))!=NULL) {
            if(strcmp(entry->d_name, ".")!=0 && entry->d_name[0]!='.' && entry->d_type!=4){
                strcat(aux, "/");
                strcat(aux,entry->d_name);
                printf("Consumidor[%d]: opção(%s)\n",consId,aux);
                if( opcao(find, aux)==1){
                    semaphore_wait(semPodeCons);
                    pthread_mutex_lock(&trinco_c);
                    correspondencias[consId][0]++;
                    pthread_mutex_unlock(&trinco_c);
                    semaphore_signal(semPodeProd);
                }
            }
        }
        closedir(dir);
    }
    pthread_exit(NULL);
```

```c
/**
 Função retorna o indice do primeiro diretório disponivel,
 se nao houver nenhum retorna -1
 */
int isEmpty(){
    int i;
    for(i=0;i<TAM && buf[i]!=NULL;i++){
        char aux[TAM]="";
        strcpy(aux, buf[i]);
        if(strcmp(aux,"0")!=0){
            return i;
        }
    }
    return -1;
}
```

```c
int opcao(FIND * find, char * dir){
    char op[TAM]="";
    int i;
    char * aux="", * current;
    current=dir;
    dir=strtok(dir, "/");
    for(i=0;dir!=NULL;i++){
        aux=dir;
        dir= strtok(NULL, "/");
    }
    for (i=0;i<TAM;i++) {
        strcpy(op,find->op[i]);
        if(op[i]=='\0') break;
        if(strcmp(op, "-name")==0){
            if(strcmp(find->procura, aux)==0){
                //printf("\n-name: procura por um ficheiro com um nome específico\n"
                printf("FOUND!!!!\n");
                return 1;
            }
        }

        else if(strcmp(op, "-iname")==0){
            if(strcasecmp(find->procura, aux)==0){
                //printf("-iname: procura por um ficheiro com um nome específico ig
                printf("FOUND!!!!\n");
                return 1;
            }
        }

        else if(strcmp(op, "-type")==0){
            //printf("-type type: procura por um tipo específico\n");
            if(strcmp(find->procura, "f")==0){
                aux=strtok(aux, ".");
                aux=strtok(NULL, ".");
                if(aux!=NULL){
                    printf("FOUND!!!\n");
                    return 1;
                }
            }
            else if(strcmp(find->procura, "d")==0){
                aux=strtok(aux, ".");
                aux=strtok(NULL, ".");
                if(aux==NULL){
                    printf("FOUND!!!\n");
                    return 1;
                }
```

- Output:

```
286         const char * argv[]={"main.c","find","WS2","-name","data.txt"};
```

```
Criei o thread produtor
Criei o thread consumidor[0]
Criei o thread consumidor[1]
Criei o thread consumidor[2]
Consumidor[0] consumiu: /Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2
Consumidor[2] consumiu: /Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2/w2ex2
Consumidor[1] consumiu: /Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2/w2ex2/data
Consumidor[2]: opção(/Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2/w2ex2/w2ex2.c)
Consumidor[1]: opção(/Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2/w2ex2/data/data.txt)
FOUND!!!!
Recebi 0 correspondencias do consumidor[0]
Recebi 1 correspondencias do consumidor[1]
Recebi 0 correspondencias do consumidor[2]
Program ended with exit code: 0
```

```
286         const char * argv[]={"main.c","find","WS2","-iname","daTa.txt"};
```

```
Criei o thread produtor
Criei o thread consumidor[0]
Criei o thread consumidor[1]
Criei o thread consumidor[2]
Consumidor[0] consumiu: /Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2
Consumidor[1] consumiu: /Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2/w2ex2
Consumidor[2] consumiu: /Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2/w2ex2/data
Consumidor[1]: opção(/Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2/w2ex2/w2ex2.c)
Recebi 0 correspondencias do consumidor[0]
Consumidor[2]: opção(/Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2/w2ex2/data/data.txt)
FOUND!!!!
Recebi 0 correspondencias do consumidor[1]
Recebi 1 correspondencias do consumidor[2]
Program ended with exit code: 0
```

```
286         const char * argv[]={"main.c","find","WS2","-type","f"};
```

```
Criei o thread produtor
Criei o thread consumidor[0]
Criei o thread consumidor[1]
Criei o thread consumidor[2]
Consumidor[0] consumiu: /Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2
Consumidor[1] consumiu: /Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2/w2ex2
Consumidor[2] consumiu: /Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2/w2ex2/data
Consumidor[2]: opção(/Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2/w2ex2/data/data.txt)
FOUND!!!
Consumidor[1]: opção(/Users/faculdade/Documents/Sistemas Operativos/projeto/MyFind/MyFind/WS2/w2ex2/w2ex2.c)
FOUND!!!
Recebi 0 correspondencias do consumidor[0]
Recebi 1 correspondencias do consumidor[1]
Recebi 1 correspondencias do consumidor[2]
Program ended with exit code: 0
```

Fase 2:Ponto d.

(25%) Devem ser criadas “n threads consumidoras”, e “n thread produtoras”. Cada tarefa produtora deve produzir diretórios para serem consumidos pelas threads consumidoras. Quando uma tarefa consumidora acaba a procura no diretório corrente deve consultar se existe mais diretórios para consumir. Quando todas as tarefas concluírem a procura, a “main thread” deve saber quantas correspondências cada tarefa satisfez.