Universidade Fernando Pessoa

Sistemas Operativos
Trabalho Prático – Parte 2

Myfind - Pesquisa de ficheiros numa hierarquia de diretórios

```
find [onde começar][[opções][o que procurar]]
```

O comando find no UNIX é um utilitário de linha de comando para percorrer uma hierarquia de diretórios. O comando find é usado para procurar e localizar a lista de ficheiros e diretórios com base nas condições especificadas nos argumentos. O comando suporta a pesquisa por ficheiros, pasta, nome, data de criação, data de modificação, dono e permissões.

1. (100%) 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. Devem usar sincronização entre tarefas para garantir o bom funcionamento do "*myfind*".

    a. (5%) Devem fazer o *parse* dos argumentos.

    b. (25%) 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. Quando todas as tarefas concluírem a procura, a "*main thread"* deve saber quantas correspondências cada tarefa satisfez.

    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.

    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.

Opções que devem considerar:

- **-name:** procura por um ficheiro com um nome específico.
- **-iname**: procura por um ficheiro com um nome específico ignorando maiúsculas ou minúsculas.
- **-type type**: procura por um tipo específico.
- **-empty**: procura por ficheiros ou diretórios vazios.
- **-executable**: procura por ficheiros executáveis.
- **-mmin -60**: procura ficheiros modificados há *n* minutos.
- **-size +5M**: procura os ficheiros com mais ou menos que *x* tamanho (megas)

Recursos:

readdir() — Read an entry from a directory

```
#include <dirent.h>
struct dirent *readdir(DIR *dir);
```

Returns a pointer to a `dirent` structure describing the next directory entry in the directory stream associated with *dir*.

Exemplo:

```
/* This example reads the contents of a root directory.
*/

#include <dirent.h>
#include <errno.h>
#include <sys/types.h>
#include <stdio.h>

main() {
  DIR *dir;
  struct dirent *entry;

  if ((dir = opendir("/")) == NULL)
            perror("opendir() error");
  else {
          puts("contents of root:");
          while ((entry = readdir(dir)) != NULL) {
printf("               %s\n", entry->d_name);
}
    closedir(dir);
  }

}
```

Output:

```
contents of root:
  .
  ..
  bin
  dev
  etc
  lib
  tmp
  usr
```

stat() — Get file information

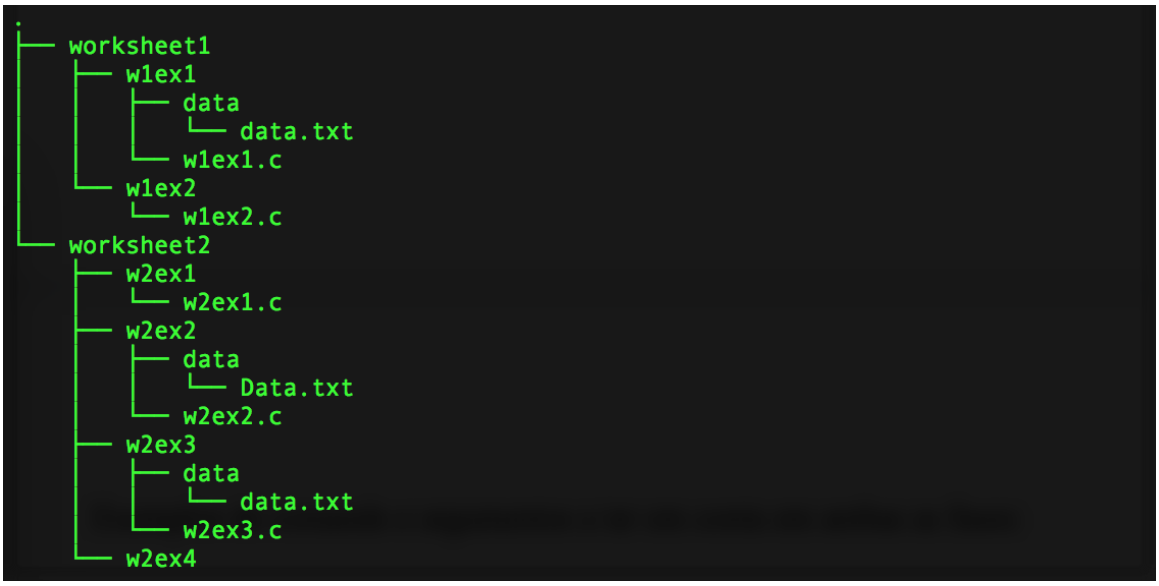
```
#include <sys/stat.h>

int stat(const char *__restrict__ pathname, struct stat
*__restrict__ info);
```

Gets status information about a specified file and places it in the area of memory pointed to by the *info* argument.

```
#include <unistd.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>

int main(int argc, char
**argv)
{
    if(argc != 2)
        return 1;

    struct stat fileStat;

if(stat(argv[1],&fileStat) <
0)
        return 1;

    printf("Information for
%s\n",argv[1]);

printf("---------------------
------\n");
printf("File Size: \t\t%d
bytes\n",fileStat.st_size);
printf("File Permissions:
\t");
printf(
(S_ISDIR(fileStat.st_mode)) ?
"d" : "-");
printf(  (fileStat.st_mode  &
S_IRUSR) ? "r" : "-");
    printf( (fileStat.st_mode
& S_IWUSR) ? "w" : "-");
    printf( (fileStat.st_mode
& S_IXUSR) ? "x" : "-");
    printf( (fileStat.st_mode
& S_IRGRP) ? "r" : "-");
    printf( (fileStat.st_mode
& S_IWGRP) ? "w" : "-");
    printf( (fileStat.st_mode
& S_IXGRP) ? "x" : "-");
    printf( (fileStat.st_mode
& S_IROTH) ? "r" : "-");
    printf( (fileStat.st_mode
& S_IWOTH) ? "w" : "-");
    printf( (fileStat.st_mode
& S_IXOTH) ? "x" : "-");
    printf("\n\n");

    return 0;
}
```

Output:

```
$ ./testProgram test.c

Information for test.c
---------------------------
File Size:              1229
bytes
Number of Links:        1
File inode:
295487
File Permissions:
-rw-r--r--
```



Exemplos de comandos e argumentos a ter em conta em ambas as fases:

1. `find . -name data.txt`

```
MBP-Andre:worksheets andrepinto$ find . -name data.txt
./worksheet1/w1ex1/data/data.txt
./worksheet2/w2ex3/data/data.txt
```

2. `find /worksheet1 -name main.c`

```
MBP-Andre:worksheets andrepinto$ find ./worksheet1 -name data.txt
./worksheet1/w1ex1/data/data.txt
```

3. `find . -name 'w2*'`

```
MBP-Andre:worksheets andrepinto$ find . -name 'w2*'
./worksheet2/w2ex1
./worksheet2/w2ex1/w2ex1.c
./worksheet2/w2ex2
./worksheet2/w2ex2/w2ex2.c
./worksheet2/w2ex3
./worksheet2/w2ex3/w2ex3.c
```

4. `find . -type f -name 'w2*'`

```
MBP-Andre:worksheets andrepinto$ find . -type f -name 'w2*'
./worksheet2/w2ex1/w2ex1.c
./worksheet2/w2ex2/w2ex2.c
./worksheet2/w2ex3/w2ex3.c
```

5. `find /home -iname Data.txt`

```
MBP-Andre:worksheets andrepinto$ find . -iname data.txt
./worksheet1/w1ex1/data/data.txt
./worksheet2/w2ex2/data/Data.txt
./worksheet2/w2ex3/data/data.txt
```

6. `find . -type d -name worksheet`

```
MBP-Andre:worksheets andrepinto$ find . -type d -name worksheet
MBP-Andre:worksheets andrepinto$
```

7. find . -type d -name 'worksheet*'

```
MBP-Andre:worksheets andrepinto$ find . -type d -name 'worksheet*'
./worksheet1
./worksheet2
```

8. find . -type f -name '*.c'

```
MBP-Andre:worksheets andrepinto$ find . -type f -name '*.c'
./worksheet1/w1ex1/w1ex1.c
./worksheet1/w1ex2/w1ex2.c
./worksheet2/w2ex1/w2ex1.c
./worksheet2/w2ex2/w2ex2.c
./worksheet2/w2ex3/w2ex3.c
```

9. find . -type d -empty

```
MBP-Andre:worksheets andrepinto$ find . -type d -empty
./worksheet2/w2ex4
```

10. find . -mmin -6

```
MBP-Andre:worksheets andrepinto$ find . -mmin -6
./.DS_Store
./worksheet2
./worksheet2/.DS_Store
./worksheet2/w2ex4
```

11. find . -size +10M

```
MBP-Andre:worksheets andrepinto$ find . -size +10M
./worksheet1/w1ex1/data/data.txt
```