Технически университет - София
Факултет по приложна математика и информатика
Учебна дисциплина “Операционни системи”

Курсов проект

Тема: Имплементация на системната функция “**ls**” с възможни аргументи “**-l**”, “**-A**” и “**-R**” на програмния език C

Изработил:

Огнян Барух
Фак. №: 471221021

Научен ръководител:

доц. д-р Симеон Цветанов

СОФИЯ
2 0 2 5

СЪДЪРЖАНИЕ

УВОД

Unix разполага с множество системни функции, които потребителят може да извика чрез конзолата, които да изпълнят определена работа. Такава функция е “`ls`”, която изброява файловете и директориите, намиращи се в пътищата, зададени като аргументи на функцията. Също така тя предоставя възможността да се използват множество флагови аргументи за сортиране на резултата, за специален формат на резултата, за филтриране на резултата и др. Най-популярните сред тях са “`-l`” - за подробна информация за всеки файл и директория, “`-A`” - за извеждане и на скритите файлове и директории и “`-R`” - за рекурсивна обработка на всички поддиректории, намиращи се под пътя, зададен като аргумент на програмата.

Целта на курсовата работа е да се имплементира системната функция “`ls`” с флагови аргументи “`-l`”, “`-A`” и “`-R`”, като потребителят трябва да може да избере да извика програмата с един или няколко от флаговите аргументи, както и да подаде нула, един или няколко пътища, които да бъдат обработени от програмата. Изхода на програмата трябва да наподобява изхода на системната функция “`ls`” и трябва да се грижи да информира потребителя за възникнали грешки по време на изпълнението си.

ГЛАВА I. ИЗИСКВАНИЯ КЪМ КУРСОВАТА РАБОТА

1.1 Основни изисквания към приложението

- Да се разработи приложение, което имплементира системната функция “`ls`” на програмния език C.
- Приложението трябва да поддържа подаването на следните аргументи при конзолно стартиране на програмата: “`-l`” (за подробна информация за изброените файлове и директории), “`-A`” (за изброяване и на скрити файлове и директории), “`-R`” (за рекурсивно изброяване на файловете във всички поддиректории).
- Приложението трябва да поддържа подаването на аргументите поотделно и заедно.
- Приложението трябва да приема произволен брой аргументи след “`-l`”, “`-A`” и “`-R`”[1], които да бъдат файлове или директории, върху които да се изпълни програмата за изброяване на файлове.

1.2 Изисквания към програмата при подаден аргумент “-l”

- Програмата трябва да изпише в конзолата подробна информация за съответния файл, като трябва да спазва форматът, използван от системната функция “`ls`” в UNIX[2]:
    - Преди изброяването на файловете в директорията се изписва бройката блокове в паметта, задалени за файловете в съответната директория:
      `total 16`
    - За всеки файл на един ред се изписва един ред с разширената информация за съответния файл.
    - Изписва се типа на файла с един символ[3]:
      `d`
      Използват се следните символи:
      “`-`” - за обикновени файлове - текстови, изображения, др.
      “`d`” - за директории
      “`c`” - за специални файлове, обслужващи устройства, които изпращат информация символ по символ
      “`b`” - за специални файлове, обслужващи устройства, които изпращат информация на блокове
      “`p`” - за пайпове

“`l`” - за линкове
“`s`” - за сокети

- Без отстояние се изписват правата на собственика на файла, на потребителите в групата, в която е собственикът на файла и всички останали потребители. Ако правата са налични се използват символите “`r`” за права за четене, “`w`” за права за писане и “`x`” за права за изпълнение. Ако правата не са налични, се изписва “`-`”:
  `rwxr--r--`
- С един символ отстояние се изписва броят на символичните линкове към съответния файл:
  `1`
- С един символ отстояние се изписва името на собственика на съответния файл:
  `user-name`
- С един символ отстояние се изписва името на групата, в която е част собственика на съответния файл:
  `group-name`
- С един символ отстояние се изписва размерът на съответния файл в байтове:
  `2048`
- С един символ отстояние се изписва датата и часа, когато последно е променян съответния файл:
  `Jan 1 10:00`
- С един символ отстояние се изписва името на съответния файл:
  `main.c`

1.3 Изисквания към програмата при подаден аргумент “-A”

- Програмата трябва да изпише в конзолата всички файлове (открити и скрити) с изключение на текущата директорията и родителската директория.

1.4 Изисквания към програмата при подаден аргумент “-R”

- Програмата трябва да изброи файловете във всички поддиректории на директорията, подадена на програмата.
- В началото се изброяват файловете в директорията, подадена на програмата.
- С един нов ред отстояние се изброяват рекурсивно файловете във всяка поддиректория, изброени по азбучен ред.

1.5 Изисквания за изпълнение и тестване на приложението

- Приложението трябва да предостави “`Makefile`”[4], чрез който да се компилира програмата и чрез който да могат да бъдат изтрити всички компилационни файлове.
- Приложението трябва да предостави списък от файлове и директории (скрити и открити) с различен размер и различни права за достъп, които да се използват за тестване.
- Приложението трябва да бъде тествано с различните аргументи, както и с различни файлове и директории и да се сравни с изхода на системната функция “`ls`” със същите аргументи.

ГЛАВА II. ПРОЕКТИРАНЕ НА КУРСОВАТА РАБОТА

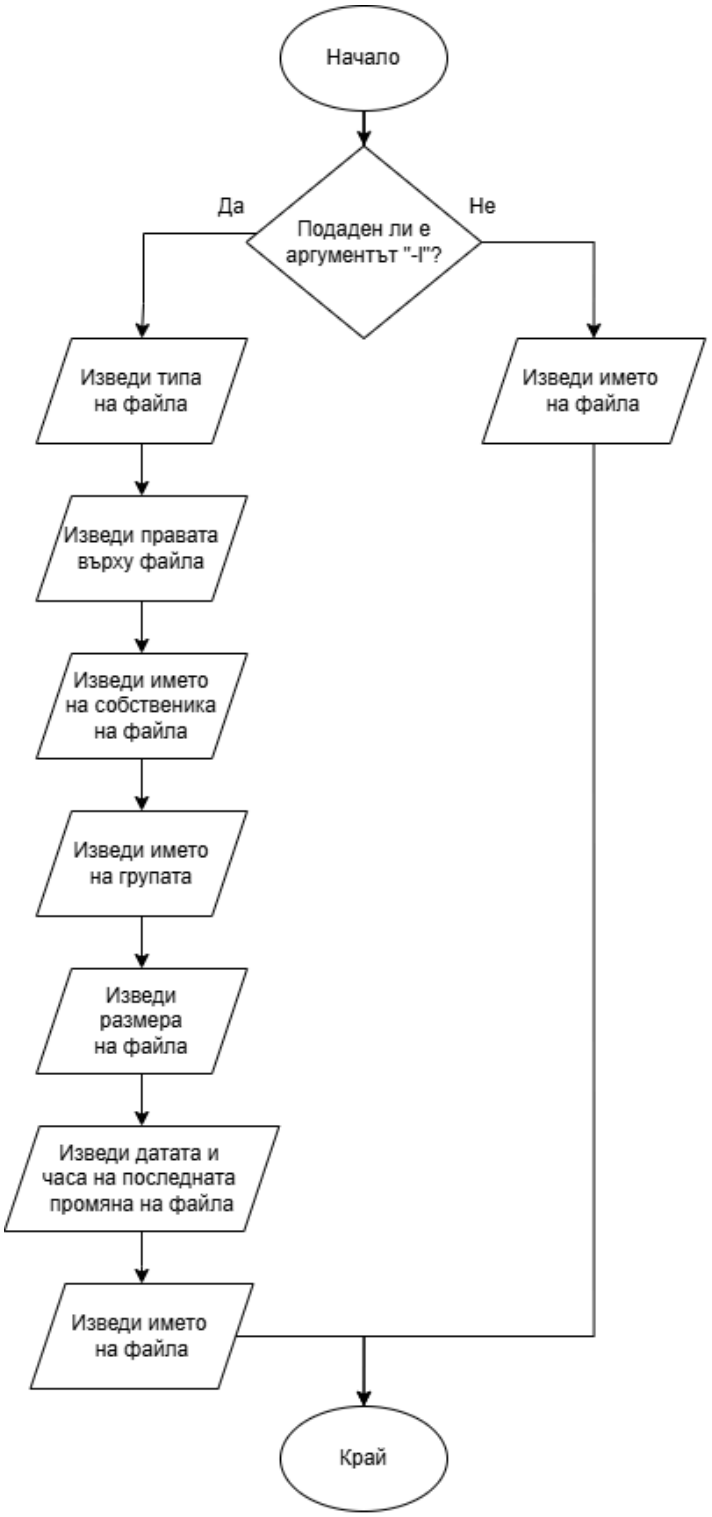
2.1 Основен ход на програмата

При стартирането си програмата приема и обработва флаговите аргументи, както и аргументите, върху които програмата ще работи. Впоследствие се обхождат файловете и директориите, подадени към програмата, и се обработват в зависимост от подадените флагове. Блоковата схема, описваща основния ход на програмата, е изобразена във *Фиг. 2.1*.

Фиг. 2.1: Основен ход на програмата

2.2 Ход на програмата при подаден аргумент “-l”

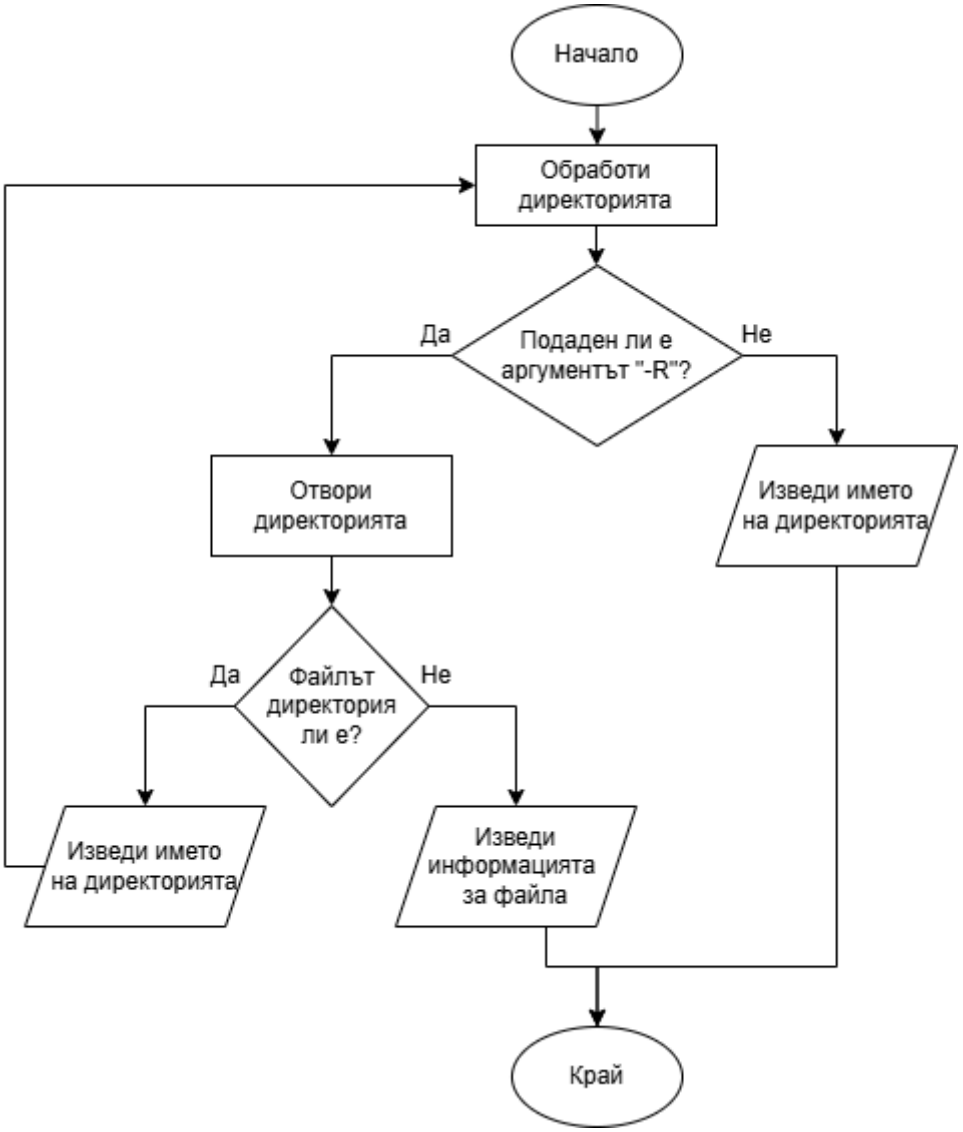
При подаден аргумент “`-l`”, за всеки файл трябва да бъде изведена разширена информация. Блоковата схема за хода на обработката на файл при подаден аргумент “`-l`” е изобразена във *Фиг. 2.2*.



Фиг. 2.2: Ход на обработката на файл при подаден аргумент “-l”

2.3 Ход на програмата при подаден аргумент “-R”

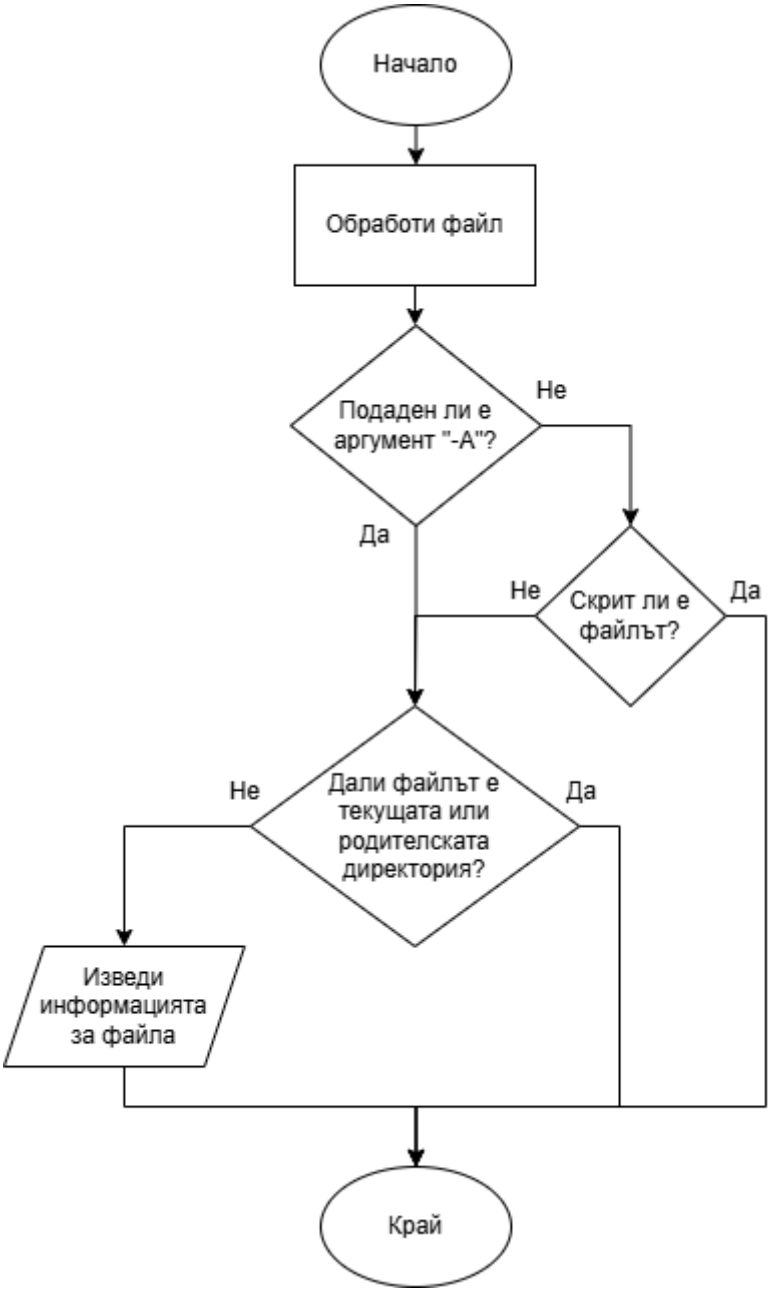
При подаден аргумент “`-R`” всяка директория трябва да бъде изведена и впоследствие обходена и обработена. Блоковата схема за хода на обработка на директория при подаден аргумент “`-R`” е изобразена във *Фиг. 2.3*.



Фиг. 2.3: Ход на обработката на файл при подаден аргумент “-R”

2.4 Ход на програмата при подаден аргумент “-A”

При подаден аргумент “-A” програмата трябва да изведе “почти всички” файлове. Това включва скритите файлове, но не включва текущата и родителската директория. Блоковата схема за хода на обработка на файл при подаден аргумент “-A” е изобразена във *Фиг. 2.4*.



Фиг. 2.4: Ход на обработката на файл при подаден аргумент “-A”

ГЛАВА III. ПРОГРАМНА РЕАЛИЗАЦИЯ

3.1 Файлова структура на приложението

```
├── ls.c
├── helpers.c
├── helpers.h
├── Makefile
└── ls
```

3.1.1 “ls.c”

Това е основният изходен файл на програмата. Той съдържа:

- Главната функция `main`, която анализира подадените аргументи и флагове от командния ред.
- Имплементацията на функцията `ls`, която решава дали подаденият път е към директория или обикновен файл.
- Функции за обработка на файлове и директории (`process_file`, `process_directory`), които се съобразяват с подадените флагове “`-l`”, “`-A`” и “`-R`”.

3.1.2 “helpers.c”

Този файл съдържа всички помощни функции, използвани в `ls.c`, като например:

- Извличане и форматиране на информация за файлове (размер, права, дата и т.н.).
- Проверки дали дадено име е скрит файл, текуща директория, родителска директория.
- Отпечатване на грешки и поддръжка за разширените флагове.

3.1.3 “helpers.h”

Това е заглавният файл, който:

- Декларира всички функции, имплементирани в `helpers.c`, така че те да могат да се използват в `ls.c`.

- Декларира структурите от данни, като например структурата `flags_t`, използвана за съхранение на флаговете “`-l`”, “`-A`” и “`-R`”.

3.1.4 “`Makefile`”

Файлът `Makefile` описва как програмата се компилира. Той автоматизира процеса на компилация чрез цели (правила), като например:

- `all`: компилира проекта.
- `clean`: изчиства генерираните обектни файлове и изпълнимия файл.

Този файл значително улеснява работата при промени в кода и осигурява лесна поддръжка.

3.1.5 “`ls`”

Това е генерираният изпълним файл след компилиране.

3.2 Програмна реализация на помощните функции

Файлът `helpers.c` съдържа имплементацията на помощни функции, използвани за реализирането на курсовата работа. Тези функции улесняват обработката на флагове от командния ред, форматирането и извеждането на информация за файлове и директории, както и други полезни операции. По-долу са описани всички основни функции:

3.2.1 “`init_flags()`”

Тази функция създава нова структура от тип `flags_t` и я инициализира, като задава стойност 0 на всички флагове (“`a`”, “`l`”, “`r`”). Функцията заделя памет динамично и връща указател към създадената структура.

3.2.2 “`get_flags(int argc, char* const argv[], struct flags_t* flags)`”

Тази функция използва `getopt()`[5] за да анализира флаговете, подадени при изпълнение на програмата. Разпознават се следните опции:

- “`A`” - показва скрити файлове, но пропуска `.` и `..`
- “`l`” - показва детайлна информация за всеки файл
- “`R`” - рекурсивно обхождане на директории

Резултатът от анализа се запазва в структурата `flags`. Функцията връща индекса на първия аргумент, който не е флаг - т.е. директорията или файла, който трябва да се обработи.

3.2.3 “`get_type_of_file(mode_t mode)`”

Използва стойността на `mode`, върната от `stat()`[6], за да определи какъв тип е даден файл. Функцията връща символ: “`-`” - обикновен файл, “`d`” - директория, “`c`” - символно устройство, “`b`” - блоково устройство, “`p`” - пайп, “`l`” - символна връзка, “`s`” - сокет.

3.2.4 “`print_permissions(mode_t mode)`”

Извежда правата за достъп до файл под формата на девет символен низ - например `rwxr-xr--`. Това включва права за четене, писане и изпълнение за собственика, групата и останалите потребители.

3.2.5 “`get_owner_name(uid_t uid)`”

Използва системната функция `getpwuid()`[7], за да получи потребителското име, съответстващо на дадено потребителско ID. Връща указател към низ с името на собственика.

3.2.6 “`get_group_name(gid_t gid)`”

Използва системната функция `getgrgid()`[8] и връща името на групата, асоциирана с дадено групово ID.

3.2.7 “`is_current_directory(char* file)`”

Проверява дали името на файла е “`.`”, което означава текущата директория. Връща `true`, ако условието е изпълнено.

3.2.8 “`is_parent_directory(char* file)`”

Проверява дали името на файла е “`..`”, което означава родителската директория. Връща `true`, ако условието е изпълнено.

3.2.9 “`is_hidden_file(char* file)`”

Функцията проверява дали името на файл започва с точка, което го прави скрит файл в Unix/Linux. В такъв случай връща `true`.

3.2.10 “`print_partial_file_info(char* filename)`”

Извежда само името на файла. Използва се в случаите, когато не е зададен флагът “`-l`” и не се изисква детайлна информация.

3.2.11 “`print_full_file_info(char* path_to_file, char* filename)`”

Събира и извежда пълна информация за файла. Използва се в случаите, когато е зададен флагът “`-l`”. Използва `stat()`[6], за да получи:

- типа на файла
- правата за достъп
- броя на линковете
- името на собственика
- името на групата
- размера на файла
- последно време на промяна
- името на файла

3.2.12 “`get_file_block_count(char* path_to_file)`”

Използва `stat()`[6] за да получи броя на блоковете, заделени за файла (в 512-байтови единици) и ги преобразува в 1K блокове чрез деление на 2.

3.2.13 “`get_total_block_count(char* directory, struct flags_t* flags`)”

Обхожда всички файлове в дадена директория и сумира броя на блоковете за тези, които трябва да се показват (в зависимост от флаговете). Пропускат се текущата и родителската директория, както и скрити файлове, ако не е зададен флагът “`-A`”.

3.2.14 “`print_directory_error_message(char* directory)`”

Използва `perror()`[9], за да изведе грешка, ако дадена директория не може да бъде отворена (например поради липса на права или ако не съществува).

3.2.15 “`print_file_error_message(char* file)`”

Използва `perror()`[9], за да изведе грешка, ако даден файл не може да бъде отворен.

3.2.16 “`has_subdirectories(char* directory, struct flags_t* flags)`”

Отваря дадена директория и проверява дали в нея се съдържа поне една поддиректория. Пропуска текущата и родителската директория, както и скрити файлове, ако не е зададен флагът “`-A`”.

3.3 Програмна реализация на основния програмен файл

Файлът `ls.c` съдържа имплементацията на основните функции, използвани за реализирането на курсовата работа:

3.3.1 “`void process_file(char* path_to_file, char* filename, struct flags_t* flags)`”

Тази функция обработва отделен файл. Проверява съществуването на файла, дали трябва да бъде показан в зависимост от подадените флагове (“`-A`”, “`-l`”), и отпечатва подходящата информация. Скритите файлове се игнорират, освен ако не е подаден флагът “`-A`”. Ако файлът не съществува, се извежда съобщение за грешка.

3.3.2 “`void process_directory(char* directory, struct flags_t* flags)`”

Тази функция се грижи за обработването на цяла директория. Ако е подаден флаг “`-l`”, тя изчислява общия брой блокове, които файловете заемат и го отпечатва. Обхожда се всяко съдържание на директорията и се обработва с помощта на `process_file()`.

Ако е подаден флагът “`-R`”, програмата събира всички поддиректории и след първоначалния преглед ги обхожда рекурсивно чрез извикване на `ls()` за всяка една от тях.

3.3.3 “`void ls(char* path_to_file, struct flags_t* flags)`”

Функцията `ls()` проверява дали подаденият път води до обикновен файл или директория. Ако е файл, той се обработва с `process_file()`. Ако е директория - с `process_directory()`. Това е централната функция, която се използва както от `main()`, така и рекурсивно при обход на поддиректории.

3.3.4 “`int main(int argc, char *argv[])`”

Главната функция на програмата:

- Инициализира структурата с флагове.
- Извиква `get_flags()` за обработка на подадените опции от командния ред.
- В зависимост от броя на аргументите:
  - Ако няма аргументи, се извиква `ls()` върху текущата директория.
  - Ако има един аргумент, се проверява дали е файл или директория, и се обработва съответно.
  - Ако има няколко аргумента, първо се обработват обикновените файлове, а след това - директориите. За всяка директория се извежда заглавие (името на директорията), след което се извиква `ls()`.

Накрая се освобождава заделената памет за флаговете.

3.4 Компилиране и изпълнение на приложението

Използва се т.нар. `Makefile`, който автоматизира компилирането на курсовата работа, използвайки двата основни изходни файла: `ls.c` и `helpers.c`. `Makefile`-ът дефинира правилата за компилиране, свързване и почистване на проекта. Описани са следните правила:

3.4.1 Основно правило

```
all: helpers.o ls.o
    gcc -o ls helpers.o ls.o
```

Това е основната цел, която се изпълнява по подразбиране, когато се напише само `make` в терминала. Първо се компилират обектните файлове `helpers.o` и `ls.o`, а след това те се свързват заедно с помощта на `gcc`, за да се създаде изпълнимият файл `ls`.

3.4.2 Правило за компилиране на обектния файл `helpers.o`

```
helpers.o: helpers.h helpers.c
    gcc -c helpers.c
```

Тази цел указва как да се създаде обектният файл `helpers.o`. Зависимостите са заглавният файл `helpers.h` и изходният код `helpers.c`.

3.4.3 Правило за компилиране на обектния файл `ls.o`

```
ls.o: ls.c
    gcc -c ls.c
```

Тази цел указва как да се създаде обектният файл `ls.o`. Зависимостите са заглавният файл `ls.h` и изходният код `ls.c`.

3.4.4 Правило за изчистване на работната директория

```
rm *.o ls
```

Това е допълнителна помощна цел, която изтрива всички обектни файлове (`*.o`) и изпълнимия файл `ls`. Използва се, когато потребителят иска да „изчисти“ проекта и да започне компилацията отначало. Изпълнява се с командата `make clean`.

ГЛАВА IV. ЕКСПЕРИМЕНТ И АНАЛИЗ

4.1 Създаване на файлове и директории за тестване

Създадена е следната структура за тестване на курсовата работа:

```
data
├── dir-1
│   ├── dir-1-file-1.txt
│   └── .dir-1-hidden-file.txt
├── file-1.txt
├── file-2.txt
└── .hidden-file.txt
```

4.2 Тестване на “ls” без аргументи

4.2.1 Тестване с текущата директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls
data  helpers.c  helpers.h  helpers.o  ls  ls.c  ls.o  makefile
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls
data
helpers.c
helpers.h
helpers.o
ls
ls.c
ls.o
makefile
```

4.2.2 Тестване с един файл

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls makefile
makefile
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls makefile
makefile
```

4.2.3 Тестване с една директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls data/
dir-1  file-1.txt  file-2.txt
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls data/
dir-1
file-1.txt
file-2.txt
```

4.2.4 Тестване с един файл и една директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls makefile data/
makefile

data/:
dir-1  file-1.txt  file-2.txt
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls makefile data/
makefile

data/:
dir-1
file-1.txt
file-2.txt
```

4.3 Тестване на “ls” с аргумент “-l”

4.3.1 Тестване с текущата директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls -l
total 68
drwxrwxrwx 1 obaruh obaruh   512 Apr 12 16:24 data
-rwxrwxrwx 1 obaruh obaruh  7543 Apr 12 16:24 helpers.c
-rwxrwxrwx 1 obaruh obaruh  6322 Apr 12 16:24 helpers.h
-rwxrwxrwx 1 obaruh obaruh  8528 Apr 12 16:24 helpers.o
-rwxrwxrwx 1 obaruh obaruh 22000 Apr 12 16:24 ls
-rwxrwxrwx 1 obaruh obaruh  7838 Apr 12 16:24 ls.c
-rwxrwxrwx 1 obaruh obaruh  7112 Apr 12 16:24 ls.o
-rwxrwxrwx 1 obaruh obaruh   140 Apr 12 16:24 makefile
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls -l
total 68
drwxrwxrwx 1 obaruh obaruh 512 Apr 12 16:24 data
-rwxrwxrwx 1 obaruh obaruh 7543 Apr 12 16:24 helpers.c
-rwxrwxrwx 1 obaruh obaruh 6322 Apr 12 16:24 helpers.h
-rwxrwxrwx 1 obaruh obaruh 8528 Apr 12 16:24 helpers.o
-rwxrwxrwx 1 obaruh obaruh 22000 Apr 12 16:24 ls
-rwxrwxrwx 1 obaruh obaruh 7838 Apr 12 16:24 ls.c
-rwxrwxrwx 1 obaruh obaruh 7112 Apr 12 16:24 ls.o
-rwxrwxrwx 1 obaruh obaruh 140 Apr 12 16:24 makefile
```

4.3.2 Тестване с един файл

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls -l makefile
-rwxrwxrwx 1 obaruh obaruh 140 Apr 12 16:24 makefile
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls -l makefile
-rwxrwxrwx 1 obaruh obaruh 140 Apr 12 16:24 makefile
```

4.3.3 Тестване с една директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls -l data/
total 0
drwxrwxrwx 1 obaruh obaruh 512 Apr 12 16:24 dir-1
-rwxrwxrwx 1 obaruh obaruh  21 Apr 12 16:24 file-1.txt
-rwxrwxrwx 1 obaruh obaruh  27 Apr 12 16:24 file-2.txt
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls -l data/
total 0
drwxrwxrwx 1 obaruh obaruh 512 Apr 12 16:24 dir-1
-rwxrwxrwx 1 obaruh obaruh 21 Apr 12 16:24 file-1.txt
-rwxrwxrwx 1 obaruh obaruh 27 Apr 12 16:24 file-2.txt
```

4.3.4 Тестване с един файл и една директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls -l makefile data/
-rwxrwxrwx 1 obaruh obaruh 140 Apr 12 16:24 makefile

data/:
total 0
drwxrwxrwx 1 obaruh obaruh 512 Apr 12 16:24 dir-1
-rwxrwxrwx 1 obaruh obaruh  21 Apr 12 16:24 file-1.txt
-rwxrwxrwx 1 obaruh obaruh  27 Apr 12 16:24 file-2.txt
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls -l makefile data/
-rwxrwxrwx 1 obaruh obaruh 140 Apr 12 16:24 makefile

data/:
total 0
drwxrwxrwx 1 obaruh obaruh 512 Apr 12 16:24 dir-1
-rwxrwxrwx 1 obaruh obaruh 21 Apr 12 16:24 file-1.txt
-rwxrwxrwx 1 obaruh obaruh 27 Apr 12 16:24 file-2.txt
```

4.4 Тестване на “ls” с аргумент “-A”

4.4.1 Тестване с текущата директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems/data$ ls -A
.hidden-file.txt  dir-1  file-1.txt  file-2.txt
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems/data$ ../ls -A
.hidden-file.txt
dir-1
file-1.txt
file-2.txt
```

4.4.2 Тестване с един файл

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems/data$ ls -A .hidden-file.txt
.hidden-file.txt
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems/data$ ../ls -A .hidden-file.txt
.hidden-file.txt
```

4.4.3 Тестване с една директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems/data$ ls -A dir-1/
.dir-1-hidden-file.txt  dir-1-file-1.txt
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems/data$ ../ls -A dir-1/
.dir-1-hidden-file.txt
dir-1-file-1.txt
```

4.4.4 Тестване с един файл и една директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems/data$ ls -A .hidden-file.txt dir-1/
.hidden-file.txt

dir-1/:
.dir-1-hidden-file.txt  dir-1-file-1.txt
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems/data$ ../ls -A .hidden-file.txt dir-1/
.hidden-file.txt

dir-1/:
.dir-1-hidden-file.txt
dir-1-file-1.txt
```

4.5 Тестване на “ls” с аргумент “-R”

4.5.1 Тестване с текущата директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls -R
.:
data  helpers.c  helpers.h  helpers.o  ls  ls.c  ls.o  makefile

./data:
dir-1  file-1.txt  file-2.txt

./data/dir-1:
dir-1-file-1.txt
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls -R
.:
data
helpers.c
helpers.h
helpers.o
ls
ls.c
ls.o
makefile

./data:
dir-1
file-1.txt
file-2.txt

./data/dir-1:
dir-1-file-1.txt
```

4.5.2 Тестване с един файл

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls -R makefile
makefile
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls -R makefile
makefile
```

4.5.3 Тестване с една директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls -R data/
data/:
dir-1  file-1.txt  file-2.txt

data/dir-1:
dir-1-file-1.txt
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls -R data/
data/:
dir-1
file-1.txt
file-2.txt

data/dir-1:
dir-1-file-1.txt
```

4.5.4 Тестване с един файл и една директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls -R makefile data/
makefile

data/:
dir-1  file-1.txt  file-2.txt

data/dir-1:
dir-1-file-1.txt
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls -R makefile data/
makefile

data/:
dir-1
file-1.txt
file-2.txt

data/dir-1:
dir-1-file-1.txt
```

4.6 Тестване на “ls” с всички аргументи

4.6.1 Тестване с текущата директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls -AlR
.:
total 68
drwxrwxrwx 1 obaruh obaruh   512 Apr 12 16:24 data
-rwxrwxrwx 1 obaruh obaruh  7543 Apr 12 16:24 helpers.c
-rwxrwxrwx 1 obaruh obaruh  6322 Apr 12 16:24 helpers.h
-rwxrwxrwx 1 obaruh obaruh  8528 Apr 12 16:24 helpers.o
-rwxrwxrwx 1 obaruh obaruh 22000 Apr 12 16:24 ls
-rwxrwxrwx 1 obaruh obaruh  7838 Apr 12 16:24 ls.c
-rwxrwxrwx 1 obaruh obaruh  7112 Apr 12 16:24 ls.o
-rwxrwxrwx 1 obaruh obaruh   140 Apr 12 16:24 makefile

./data:
total 0
-rwxrwxrwx 1 obaruh obaruh  23 Apr 12 16:24 .hidden-file.txt
drwxrwxrwx 1 obaruh obaruh 512 Apr 12 16:24 dir-1
-rwxrwxrwx 1 obaruh obaruh  21 Apr 12 16:24 file-1.txt
-rwxrwxrwx 1 obaruh obaruh  27 Apr 12 16:24 file-2.txt

./data/dir-1:
total 0
-rwxrwxrwx 1 obaruh obaruh 35 Apr 12 16:24 .dir-1-hidden-file.txt
-rwxrwxrwx 1 obaruh obaruh 28 Apr 12 16:24 dir-1-file-1.txt
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls -AlR
.:
total 68
drwxrwxrwx 1 obaruh obaruh 512 Apr 12 16:24 data
-rwxrwxrwx 1 obaruh obaruh 7543 Apr 12 16:24 helpers.c
-rwxrwxrwx 1 obaruh obaruh 6322 Apr 12 16:24 helpers.h
-rwxrwxrwx 1 obaruh obaruh 8528 Apr 12 16:24 helpers.o
-rwxrwxrwx 1 obaruh obaruh 22000 Apr 12 16:24 ls
-rwxrwxrwx 1 obaruh obaruh 7838 Apr 12 16:24 ls.c
-rwxrwxrwx 1 obaruh obaruh 7112 Apr 12 16:24 ls.o
-rwxrwxrwx 1 obaruh obaruh 140 Apr 12 16:24 makefile

./data:
total 0
-rwxrwxrwx 1 obaruh obaruh 23 Apr 12 16:24 .hidden-file.txt
drwxrwxrwx 1 obaruh obaruh 512 Apr 12 16:24 dir-1
-rwxrwxrwx 1 obaruh obaruh 21 Apr 12 16:24 file-1.txt
-rwxrwxrwx 1 obaruh obaruh 27 Apr 12 16:24 file-2.txt

./data/dir-1:
total 0
-rwxrwxrwx 1 obaruh obaruh 35 Apr 12 16:24 .dir-1-hidden-file.txt
-rwxrwxrwx 1 obaruh obaruh 28 Apr 12 16:24 dir-1-file-1.txt
```

4.6.2 Тестване с един файл

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls -AlR makefile
-rwxrwxrwx 1 obaruh obaruh 140 Apr 12 16:24 makefile
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls -AlR makefile
-rwxrwxrwx 1 obaruh obaruh 140 Apr 12 16:24 makefile
```

4.6.3 Тестване с една директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls -AlR data/
data/:
total 0
-rwxrwxrwx 1 obaruh obaruh  23 Apr 12 16:24 .hidden-file.txt
drwxrwxrwx 1 obaruh obaruh 512 Apr 12 16:24 dir-1
-rwxrwxrwx 1 obaruh obaruh  21 Apr 12 16:24 file-1.txt
-rwxrwxrwx 1 obaruh obaruh  27 Apr 12 16:24 file-2.txt

data/dir-1:
total 0
-rwxrwxrwx 1 obaruh obaruh 35 Apr 12 16:24 .dir-1-hidden-file.txt
-rwxrwxrwx 1 obaruh obaruh 28 Apr 12 16:24 dir-1-file-1.txt
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls -AlR data/
data/:
total 0
-rwxrwxrwx 1 obaruh obaruh 23 Apr 12 16:24 .hidden-file.txt
drwxrwxrwx 1 obaruh obaruh 512 Apr 12 16:24 dir-1
-rwxrwxrwx 1 obaruh obaruh 21 Apr 12 16:24 file-1.txt
-rwxrwxrwx 1 obaruh obaruh 27 Apr 12 16:24 file-2.txt

data/dir-1:
total 0
-rwxrwxrwx 1 obaruh obaruh 35 Apr 12 16:24 .dir-1-hidden-file.txt
-rwxrwxrwx 1 obaruh obaruh 28 Apr 12 16:24 dir-1-file-1.txt
```

4.6.4 Тестване с един файл и една директория

```
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ls -AlR makefile data/
-rwxrwxrwx 1 obaruh obaruh 140 Apr 12 16:24 makefile

data/:
total 0
-rwxrwxrwx 1 obaruh obaruh  23 Apr 12 16:24 .hidden-file.txt
drwxrwxrwx 1 obaruh obaruh 512 Apr 12 16:24 dir-1
-rwxrwxrwx 1 obaruh obaruh  21 Apr 12 16:24 file-1.txt
-rwxrwxrwx 1 obaruh obaruh  27 Apr 12 16:24 file-2.txt

data/dir-1:
total 0
-rwxrwxrwx 1 obaruh obaruh 35 Apr 12 16:24 .dir-1-hidden-file.txt
-rwxrwxrwx 1 obaruh obaruh 28 Apr 12 16:24 dir-1-file-1.txt
obaruh@TUF-NA:/mnt/d/Coding/University/6_Semester/Operating-Systems$ ./ls -AlR makefile data/
-rwxrwxrwx 1 obaruh obaruh 140 Apr 12 16:24 makefile

data/:
total 0
-rwxrwxrwx 1 obaruh obaruh 23 Apr 12 16:24 .hidden-file.txt
drwxrwxrwx 1 obaruh obaruh 512 Apr 12 16:24 dir-1
-rwxrwxrwx 1 obaruh obaruh 21 Apr 12 16:24 file-1.txt
-rwxrwxrwx 1 obaruh obaruh 27 Apr 12 16:24 file-2.txt

data/dir-1:
total 0
-rwxrwxrwx 1 obaruh obaruh 35 Apr 12 16:24 .dir-1-hidden-file.txt
-rwxrwxrwx 1 obaruh obaruh 28 Apr 12 16:24 dir-1-file-1.txt
```

ЗАКЛЮЧЕНИЕ

С оглед на извършените експерименти и получените резултати, може да се заключи, че изискванията, поставени върху курсовата работа са изпълнени успешно. Потребителят има възможността да изброи множество файлове и директории, използвайки три от най-полезните флагови аргументи: “`-l`” - за извеждане на подробна информация за всеки файл, “`-A`” - за извеждане на скритите файлове и “`-R`” - за рекурсивно обхождане на всички поддиректории и извеждането на техните файлове.

Бъдещото развитие на програмата включва добавянето на други флагови аргументи на системната функция “`ls`” - “`-a`” - за извеждане на скритите файлове, както и на текущата и родителската директория, “`-T`”, “`-s`” или “`-X`” за извеждането на файлове със специфична подредба.

ИЗТОЧНИЦИ

1. “`ls`” manual - https://man7.org/linux/man-pages/man1/ls.1.html
2. Understanding of “`ls`” command with a long listing format output - https://linuxconfig.org/understanding-of-ls-command-with-a-long-listing-format-output-with-permission-bits
3. File types - https://www.gnu.org/software/libc/manual/html_node/Testing-File-Type.html
4. `Makefile` tutorial - https://makefiletutorial.com
5. “`getopt`” manual - https://www.man7.org/linux/man-pages/man3/getopt.3.html
6. “stat” manual - https://man7.org/linux/man-pages/man2/stat.2.html
7. “`getpwuid`” manual - https://man7.org/linux/man-pages/man3/getpwuid.3p.html
8. “`getgrgid`” manual - https://man7.org/linux/man-pages/man3/getgrgid.3p.html
9. “`perror`” manual - https://man7.org/linux/man-pages/man3/perror.3.html