Технологично училище "Електронни системи"
Технически университет – София

Операционни системи

Задача 4: Реализация на `ls`

elsys.os.2014@gmail.com

2 юни 2015 г.

1 Условие на задачата

1.1 Основна функционалност (20 точки)

Целта на задачате е да се реализира стандартната `UNIX` комадна `ls` с добавена функционалност.

Командата `ls` се използва за извеждане на съдържанието на директории. От командния ред програмата получава списък от имена на файлове и/или директории, а на стандартния изход извежда информация за тях. Ако подаденото име, е съществуващ файл, се извежда името му и неговия тип, а ако името е на директория - имената и типовете на всички файлове, съдържащи се в директорията.

Прието е типовете на файловете в POSIX да се означават по следния начин:

- обикновен файл - '-'
- директория - 'd'
- блоково устройство - 'b'
- символно устройство - 'c'
- програмен поток - 'p'
- символна връзка - 'l'
- сокет - 's'

Примерен изход на програмата:

```
$ ls a.txt
- a.txt
$ ls /usr/share/texmf/tex
- fontinst
- generic
d latex
```

Забележка: `ls` не показва скрити файлове.

В случай на повече от един аргумент изходът изглежда по следния начин:

```
$ ls a.txt b.txt /usr/share/texmf/tex /usr/share/texmf/metapost/
- a.txt
- b.txt

/usr/share/texmf/metapost/:
- metauml

/usr/share/texmf/tex:
- fontinst
- generic
d latex
```

1.2 Поддръжка на `-a` (20 точки)

Командата `ls` трябва да поддържа незадължителния аргумент `-a`.

При използване на опцията `-a` командата трябва да извежда и скритите файлове. Например:

```
$ ls -a
d .
d ..
```

```
- a.out
- a.txt
- b.txt
d forbidden
- .hidden
- main.c
```

1.3 Поддръжка на `-l` (20 точки)

Командата `ls` трябва да поддържа незадължителния аргумент `-l`.

При използване на опцията `-l` командата трябва да извежда подробна информация за файловете. Например:

```
$ ls -l
total 28
-rwxrwxr-x 1 hristo hristo 8510 май 31 01:56 a.out
-rw-rw-r-- 1 hristo hristo   13 май 31 01:51 a.txt
-rw-rw-r-- 1 hristo hristo   27 май 31 01:51 b.txt
d--------- 2 hristo hristo 4096 май 31 01:54 forbidden
-rw-rw-r-- 1 hristo hristo   72 май 31 01:51 main.c
```

1.4 Поддръжка на `-R` (20 точки)

Командата `ls` трябва да поддържа незадължителния аргумент `-R`.

При използване на опцията `-R` командата трябва при обработване на директории да ги обхожда рекурсивно. При примерна файлова структура:

```
.
|-- a.out
|-- a.txt
|-- b.txt
|-- dir1
|   |-- dir11
|   |-- file01.txt
|-- dir2
|-- dir3
```

резултатът от изпълнението на програмата трябва да е:

```
$ ls -R

.:
- a.out
- a.txt
- b.txt
d dir1
d dir2
d dir3

./dir1:
d dir11
- file01.txt

./dir1/dir11:

./dir2:

./dir3:
```

1.5 Комбинация от опции

Възможна е произволна комбинация от опции. Например: ls -lRa, ls -al, ..

1.6 Обработка на грешки (20 точки)

Ако някой от аргументите на `ls` не е файл или директория, то програмата трябва да изведе съобщение на стандартната грешка (`stderr`). Например, ако бъдат предадени аргументи, които не са файлове, то съобщението трябва да бъде оформено по следния начин:

```
$ ls aa
ls: cannot access aa: No such file or directory
```

Ако на програмата се предаде име на директория, която потребителят няма права да отвори, то съобщението трябва да бъде оформено по следния начин:

```
$ ls forbidden/
ls: cannot open directory forbidden/: Permission denied
```

2 Изисквания към решението и оценяване

1. Решението на задачата трябва да бъде написано на езика `C` съгласно ISO/IEC 9899:1999.
2. Правилата за оценяване са следните. Приемаме, че напълно коректна и написана спрямо изискванията програма получава максималния брой точки — 100% или 70 точки. Ако в решението има пропуски, максималният брой точки ще бъде намален съгласно правилата описани по-долу.
3. Задължително към файловете с решението трябва да е приложен и `Makefile`. Изпълнимият файл, който се създава по време на компилация на решението, трябва да се казва `ls`.
4. При проверка на решението програмата ви ще бъде компилирана и тествана по следния начин:

   ```
   make
   ./ls .
   ```

   Предходната процедура ще бъде изпълнена няколко пъти с различни входни данни за да се провери дали вашата програма работи коректно.
5. Реализацията на програмата трябва да спазва точно изискванията. Всяко отклонение от изискванията ще доведе до получаване на 0 точки за съответната част от условието.
6. Работи, които са предадени по-късно от обявеното (или не са предадени), ще бъдат оценени с 0 точки.
7. Програмата ви трябва да съдържа достатъчно коментари. Оценката на решения без коментари или с недостатъчно и/или мъгляви коментари ще бъде намалена с 30%.
8. Всеки файл от решението трябва да започва със следният коментар:

```
//-------------------------------------------
// NAME: Ivan Ivanov
// CLASS: XIa
// NUMBER: 13
// PROBLEM: #1
// FILE NAME: xxxxxx.yyy.zzz (unix file name)
// FILE PURPOSE:
//  няколко реда, които описват накратко
//  предназначението на файла
//  ...
//-------------------------------------------
```

Всяка функция във вашата програма трябва да включва кратко описание в следния формат:

```
//-------------------------------------------
// FUNCTION: xxyyzz (име на функцията)
//  предназначение на функцията
// PARAMETERS:
//  списък с параметрите на функцията
//  и тяхното значение
//-------------------------------------------
```

9. Лош стил на програмиране и липсващи заглавни коментари ще ви костват 30%.

10. Програми, които не се компилират получават 0 точки. Под „не се компилират“ се има предвид произволна причина, която може да причини неуспешна компилация, включително липсващи файлове, неправилни имена на файлове, синтактични грешки, неправилен или липсващ `Makefile`, и т.н. Обърнете внимание, че в `UNIX` имената на файловете са “case sensitive”.

11. Програми, които се компилират, но не работят, не могат да получат повече от 50%. Под „компилира се, но не работи“ се има предвид, че вие сте се опитали да решите проблема до известна степен, но не сте успели да направите пълно решение. Често срещан проблем, който спада към този случай, е че вашият Makefile генерира изпълним файл, но той е именуван с име, различно от очакваното (т.е. `ls` в разглеждания случай).

12. Безсмислени или мъгляви програми ще бъдат оценявани с 0 точки, независимо че се компилират.

13. Програми, които дават неправилни или непълни резултати, или програми, в които изходът и/или форматирането се различава от изискванията ще получат не повече от 70%.

14. Всички наказателни точки се сумират. Например, ако вашата програма няма задължителните коментари в началото на файлове и функциите се отнемат 30%, ако няма достатъчно коментари се отнемат още 30%, компилира се, но не работи правилно — още 30%, то тогава резултатът ще бъде: $70 * (100 - 30 - 30 - 30)\% = 70 * 10\% = 7$ точки

15. Работете самостоятелно. Групи от работи, които имат твърде много прилики една с друга, ще бъдат оценявани с 0 точки.