Технологично училище "Електронни системи"
Технически университет - София

Операционни системи

Задача №1: **Реализация на** tail

---

os@elsys-bg.org

6 ноември 2017 г.

1 Условие на задачата

1.1 Основна функционалност (20 точки)

Целта на задачате е да се реализира стандартната UNIX команда `tail`.

Командата `tail` се използва за показване на последните редове на файлове. От командния ред програмата получава списък от файлове и започва да ги обработва като последните 10 реда от всеки файл последователно се копират в стандартния изход (stdout), разделени със заглавна секция.

Ако файлът съдържа по-малко от 10 реда, цялото негово съдържание се копира в стандартния изход.

Ако командата не получи аргументи от командния ред, тя започва да чете от стандартния вход.

Например, ако са дадени два файла със следното съдържание `a.txt`:

```
one
two
three
four
five
six
seven
eight
nine
ten
eleven
```

и b.txt:

```
1234 1234
567
890
```

то

```
$ tail a.txt b.txt
==> a.txt <==
two
three
four
five
six
seven
eight
nine
ten
eleven

==> b.txt <==
1234 1234
567
890
```

При наличието на само един аргумент, заглавната секция се пропуска.

1.2 **Обработка на грешки** (10 **точки**)

Ако някой от аргументите на `tail` не е файл или файлът не може да се отвори, то програмата трябва да изведе съобщение на стандартната грешка (stderr). Например, ако бъдат предадени аргументи, които не са файлове

```
tail aa bb
```

съобщението трябва да бъде оформено по следния начин:

```
tail: cannot open 'aa' for reading: No such file or directory
tail: cannot open 'bb' for reading: No such file or directory
```

Ако някой от аргументите на `tail` е файл, който не може да се прочете, то програмата трябва да изведе съобщение на стандартната грешка (stderr). Например, ако бъдат предадени аргументи, които са директории

```
tail /tmp/
```

съобщението трябва да бъде оформено по следния начин:

```
tail: error reading '/tmp/': Is a directory
```

Ако `tail` не може да запише прочетеното на стандартния вход, то програмата трябва да изведе съобщение на стандартната грешка (stderr). Например, ако дискът е пълен

```
tail > /dev/full
```

съобщението трябва да бъде оформено по следния начин:

```
tail: error writing 'standard output': No space left on device
```

Ако `tail` не може да затвори успешно някой файл, то програмата трябва да изведе съобщение на стандартната грешка (stderr). Например

```
tail b.txt
```

съобщението трябва да бъде оформено по следния начин:

```
tail: error reading 'b.txt': Input/output error
```

При грешка в един файл, то изпълнението трябва да продължи към следващите (програмата не трябва да приключи).

1.3 **Поддръжка на стандартен вход** (10 **точки**)

Когато командата получи от командния ред аргумент `-`, тя трябва да интерпретира този аргумент като стандартния вход. Например командата

```
tail f1 . txt - f2 . txt
```

трябва да изпълни следната последователност от действия:

- Да копира последните 10 реда на файла f1.txt на стандартия изход.
- Да чете редове от стандартния вход, докато не бъде затворен и да копира последните 10 реда на стандартния изход.
- Да копира последните 10 реда на файла f2.txt на стандартия изход.

2 **Изисквания към решението и оценяване**

2.1 Решението на задачата трябва да бъде написано на езика C съгласно ISO/IEC 9899:2011;

2.2 Правилата за оценяване са следните. Приемаме, че напълно коректна и написана спрямо изискванията програма получава максималния брой точки - 100%.
Ако в решението има пропуски, максималният брой точки ще бъде намален съгласно правилата описани по-долу;

2.3 За работа с файлове трябва да се използва семейството от функции за работа с файлове от ниско ниво - `open()`, `close()`, `read()`, `write()`, `lseek()` и т.н. Неспазването на това изискване води до оценяване на задачата с 0 точки;

2.4 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 `tail`;

2.5 При проверка на решението програмата ви ще бъде компилирана и тествана по следния начин:

```
make
./tail .
```

Предходната процедура ще бъде изпълнена няколко пъти с различни входни данни за да се провери дали вашата програма работи коректно;

2.6 Реализацията на програмата трябва да спазва точно изискванията. Всяко отклонение от изискванията ще доведе до получаване на 0 точки за съответната част от условието;

2.7 Работи, които са предадени по-късно от обявеното (или не са предадени), ще бъдат оценени с 0 точки;

2.8 Решението ви трябва да съдържа достатъчно коментари. Оценката на решения без коментари или с недостатъчно и/или мъгляви коментари ще бъде намалена с 30%;

2.9 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 вашия изходен код 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
//--------------------------------------------------------------------
```

2.10 Лош стил на програмиране и липсващи заглавни коментари ще ви костват 30%;

2.11 Решения, 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";

2.12 Решения, 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. `tail` в разглеждания случай);

2.13 Безсмислени или мъгляви решения ще бъдат оценявани с 0 точки, независимо че се компилират;

2.14 Решения, които дават неправилни или непълни резултати, или решения, в които изходът и/или форматирането се различава от изискванията ще получат не повече от 70%;

2.15 Всички наказателни точки се сумират. Например, ако вашето решение 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: (100 - 30 - 30 - 30)% = 10%;

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