Introduction to C Programming in Unix Environment

(This lab to be done SOLO)

Task 0: Maintaining a project using `make`

You should accomplish this task **before** attending the lab session. For this task, 3 files are provided: add.s, main.c, numbers.c.

1. Login into Linux.
2. Using an editor of your choice in Linux, write a Makefile (as explained in the introduction to GNU Make Manual, see the reading material). The Makefile should provide targets for compiling the program and cleaning up the working directory.
3. Obviously, you should read all the indicated reading material before attending the lab.
4. Read puts(3) and printf(3) manuals. In what way are they different?

Tasks below to be done in lab sessions only!

If you come with tasks 1 or 2 already "done at home" you will **not be allowed to participate in the lab!** Additionally, wherever you get your code (and you are expected to write it **in the lab at lab time**, you are of course expected to understand it completely.

Task 1: The nottail program

In this task we will implement the `nottail` program:

SYNOPSIS
nottail [-n NUMBER] [-o OFILE] [-u K] **...** [FILE] **...**

DESCRIPTION
Print the first 2 lines of standard input to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, read standard input.

-n NUMBER , print first NUMBER lines instead of first 2
-o OFILE , print to OFILE instead of standard output
-u K , Print every K-th character in each line, the rest of the characters are omitted

SEE ALSO
head(1)

One can experiment with the Unix head utility before implementing nottail.

Before starting the lab you should consult the man pages for **atoi(3), strcmp(3), fprintf(3), fgetc(3), fputc(3), fopen(3), fclose(3)** to choose library functions for input and output.

In addition you are encouraged to read your program parameters in the manner of task0 main.c

Task 1a: A restricted nottail version: reading from stdin to stdout

A restricted version of nottail will be implemented, as follows:
Print the first two lines from stdin to stdout. If the flag -n NUMBER is used, print the first NUMBER lines.

NOTE: Conventionally, a line in LINUX ends with a newline character ('\n').

For example (user input in italic font, program output in normal font):
~> *nottail*
"hello"
"hello"
bye
bye
~>
NOTE that the first appearances of "hello" and bye are the input to stdin, and the second ones are the output to stdout.

Another example:
~> *nottail -n 1*
"hello"
"hello"
~>

Guidelines: First, implement the function `void rw(FILE * input, FILE * output, int n)` that reads n lines from input and prints it to output.
Next you should call rw from main with the right parameters.

Task 1b: Extending nottail: reading from files

nottail, implemented in Task 1a, should be extended to read from a file as follows:
With one FILE or more, precede each with a header giving the file name. With no FILE, read standard input.

For example:

~> echo "Mississippi\nhi\nbye\n" > file1
~> echo "Alabama\nhi\nbye\n" > file2
~> nottail -n 1 file1 file2

Will print:

file1
Mississippi

file2
Alabama

NOTE: a line in LINUX ends with a newline character ('\n') by convention.
NOTE: make sure you know how to recognize end of file.

Guideline: For the file name printing you can use fprintf(3).

Task 1c: Extending nottail with an output option

nottail, implemented in Task 1b, should be extended to writing to a file as follows:
If the flag -o OFILE is used, print to OFILE instead of standard output.

For example:

~> echo "Mississippi\nhi\nbye\n" > a.txt
~> echo "Alabama\nhi\nbye\n" > b.txt
~> nottail -o c.txt a.txt b.txt

Will create a file named c.txt with the content:
a.txt
Mississippi

b.txt
Alabama

Task 2: Extend nottail to print every #-th character in each line

nottail, implemented in Task 1c, should be extended as follows:
If the flag -u K is used, print every K-th character in each line. The rest of the characters are omitted.

For example:

~> echo "abcdef" > a.txt
~> echo "12345" >> a.txt
~> nottail -u 2 a.txt
Will print:
bdf
24

~> nottail -u 3 a.txt
Will print:
cf
3

~> nottail -u 4 a.txt
Will print:
d
4

As you can see in this last example, if more than one line should be printed (N>2), then in each line the counter of the -u flag is set to zero.

Note: you can add parameters to rw function.

Deliverables

Students who run out of time at the **end of the lab** may defer Task 2 to the completion lab with no penalty.

Page last modified on 7 March 2012, 19:54 by **achiya**
powered by **CourseWiki 2.4.2** , **gzip** supported
generated in 0.00091743 sec.