Operating Systems

Grado en Informática. Course 2018-2019

Lab assignment 0: Introduction to C programming language

To get acquainted with the C programming language we'll start to code a shell, coding of this shell will be continued in next lab assignments.

We'll start with a nearly empty shell, which is basically a loop that

- prints a *prompt*
- reads fromthe standard input a line of text which includes a command (with its arguments).
- separates the command and its arguments
- processes the comand (with its argumnets)

At this moment this *shell* has to understand only the following commands

autores [-l|-n] Prints the names and logins of the program authors. **autores -l** prints only the logins and **autores -n** prints the names

pid [-p] Prints the pid of the process executing the shell **pid -p** prints the pid of its parent process.

chdir [direct] Changes the current working directory of the shell to *direct*. When invoking without auguments it prints the current working directory of the shell

fecha Prints the current date

hora Prints the current time

fin Ends the shell

end Ends the shell

exit Ends the shell

- This program should compile cleanly (produce no warnings even when compiling with `gcc -Wall`)
- **NO RUNTIME ERROR WILL BE ALLOWED (segmentation, bus error . . .)**, unless where explicitly spcified. Programs with runtime errors will yield no score.
- This program can have no memory leaks
- When the program cannot perform its task (for whatever reason, for example, lack of privileges) it should inform the user

- All input and output is done through the standard input and output

Information on the system calls and library functions needed to code this program is available through `man`: (*printf, gets, read, write, exit, getpid, getppid, getcwd, chdir, time ...*).

WORK SUBMISSION

- Work must be done in pairs.
- The source code will be submitted to the subversion repository under a directory named **P0**
- The name of the main program file will be `p0.c`. Program must be able to be compiled with `gcc p0.c` Alternatively a `Makefile` can be supplied so that the program can be compiled with just `make`
- Only one of the members of the workgroup will submit the source code. The names and logins of all the members of the group should be in the source code of the main program (at the top of the file)

DEADLINE: This lab assignment will yield no score, neither will it be evaluated. However all the code for this assignment can be reutilized for the following assigments. This assigmente will also help get acquainted with the `svn` repository, needed for the correct submission of all of the following lab assignments (from the next assignment on, work wrongly submitted will no be evaluated)

CLUES

A shell is basically a loop

```
while (!terminado){
   imprimirPrompt();
   leerEntrada();
   procesarEntrada();
}
```

imprimirPrompt() and*leerEntrada()* can be as simple as calls to `printf` y `gets`

The first step whe processing the input string is splitting it into words. For this, the `strtok` library function comes in handy. Please notice that `strtok` nor allocates memory neither does copy strings, it just breaks the input string by inserting end of string ('\0') characters. The following function splits the string pointed by *cadena* (suposedly not null) into a NULL terminated array

of pointers (*trozos*). The function returns the number of words that were in *cadena*

```
int TrocearCadena(char * cadena, char * trozos[])
{ int i=1;

  if ((trozos[0]=strtok(cadena," \n\t"))==NULL)
    return 0;
  while ((trozos[i]=strtok(NULL," \n\t"))!=NULL)
    i++;
  return i;
}
```