**Operating Systems**   
***CS 550***

# Spy

It's nice to be able to track the activities of specific users and/or commands running on your LINUX system. Your assignment is to write a utility program called ***spy*** that enables you to do just that.

*spy* repeatedly scans the system process table, displaying processes that have started or ended since the last scan. The usage of spy is this:

***% spy [ list of patterns] [-t secs] [-n count]***

Anything enclosed in square brackets is optional. Here's what each field means:

**list of patterns:**

Is a list of patterns to look out for in the process list, possibly containing regular expressions, described briefly below. If no list is present, spy accepts all processes.

**tsecs:**

Tells ***spy*** how many seconds to wait between each scan of the system process table. The default value is one.

**count:**

Tells ***spy*** how many times the system process table should be scanned before quitting. The default value is five.

During each scan, ***spy*** produces an intermediate file containing a list of current processes in the form:

***User- id process-id command***

which it then compares with the list constructed during the last scan. The differences between the two lists indicate which processes have ended and which have begun since the last scan. This difference is then displayed, in a readable format together with the date and time.

Here's an example of such a process list:

**heydari 4286 csh spy lippke smith**   
**heydari 7949 ps -aux**

***Spy*** allows you to filter the list for patterns, possibly including regular expressions in the same form as those accepted by the LINIX utility ***grep***.

For example the command:

***% spy "^user" vi***

would consider all lines in the process list containing ***vi*** or beginning with the word ***user***. Similarly, the command:

***% spy fox heydari vi -t 10 -n 100***

would consider all lines in the process list that contain fox, heydari, or vi. ***Spy*** program would iterate 100 times at 10 second intervals.

**Error Checking:**

Your program should check for simple errors such as missing options (*ie*. % ***spy heydari -n***, where ***-n*** option is used but ***count*** is not given). If an error is detected, display an appropriate message and quit. Most utilities present a usage reminder when an ill-formed command line is entered. For example the following error message is appropriate for the following command:

***% spy -n***   
**missing count Parameter**   
**Usage: spy [list of patterns] [-t seconds] [-n count]**

***Sample Output***

Here's a sample output from the ***spy*** utility, executed from a C shell owned by heydari:   
 

***Sat Jan 30 4:15:25 EST 2012***  
***Stared: heydari 26788 csh spy chess fox -t 10***   
***Started: heydari 27258 csh spy***   
***Started: chess 1843 -csh[chess]***   
***Started: fox 24505 -csh[fox]***   
***Started: fox 24713 vi Memo***

***Sat Jan 30 4:15:35 EST 2012***  
***Ended: fox 24713 vi Memo***

***Sat Jan 30 4:15:45 EST 2012***  
***Started: fox 87253 lpr***   
***Started: Smith 76532 mail***

***Sat Jan 30 4:15:55 EST 2012***

***Sat Jan 30 4:15:65 EST 2012***

**Note:**

It is quite possible that no processes change between scans. In this case, only the time stamp is displayed, please see the last 2 output lines above.

Normally you'd execute ***spy*** in the background and redirect its output into a file.

**Handling Interrupts:**

You will want to create temporary files to hold the process lists between scans. If your shell is interrupted (by a Ctrl-C for example), you don't want these temporary files to be left in your directory, so ensure that interrupts are handled appropriately. You will find the built-in C shell command ***onintr*** or ***trap*** very useful for this purpose.