

CSIS0230A Principles of Operating Systems (Class A)

Assignment 1

Using system calls

Tutor: kmcheung@csis.hku.hk , Deadline Oct 6, 2003, 12:00pm.

In this assignment, you will practice writing C programs and using system calls. Since the aim of the assignment is to use system calls, we only allow a limited number of standard library calls.

This is a group assignment allowing 1–2 students in each group. We expect that you do the assignment in Linux. It should be impossible to complete the assignment using any other OS.

1. Task

Most users setup their computer desktop preference so that after they do not interact with the computer for some time, a screen-saver is activated. Some users even set up their desktop to lock their screen at such times. At such a time when the screen cannot be used tell the user that E-mails arrive, programs which watch over mail boxes may produce a beep sound from the computer. But if the user is away from the computer at that particular time, the user won't know that some E-mails need his attention when he gets back. One possible solution is to blink the keyboard LEDs instead.

You are to write two simple programs that checks how many E-mails are residing on a mailbox, in the “mbox” format and in the “maildir” format respectively (see below). These programs are invoked like this:

```
check_mbox /var/mail/tmchan
check_mdir Mail/new/
```

In either case, the command-line argument is the mailbox. The output of the program consists of a single line, which is the number of mails currently in the mailbox. Errors are reported through the standard error device, as well as the exit-code of the program; if there is no error the command exits with an exit code of 0, otherwise it exits with a non-zero exit-code.

After that, you are to write a program `mail_led` that periodically run a command to check the number of E-mails in the mailbox of the user. It is invoked with arguments which specify the command to execute and the command-line argument(s) to use. For example,

```
mail_led check_mbox /var/mail/tmchan
```

In this example, `mail_led` invokes the `check_mbox Mail/tmchan` periodically in order to check for E-mails. Then it inspects the output of the `check_mbox` program to find the number of mails in the mailbox. After each check, the keyboard LED should be blinking if there is mail, and should be off if there is no mail. The rate of blinking indicates how many mails are waiting in the mailbox: if there are n mails in the mailbox, the keyboard LED should change state approximately n times a second.

2. Requirements

- Your programs should work both in a text-mode console or under X. To do so, `mail_led` should attempt to use `ioctl()` on the standard output device to change the keyboard LED for `scroll-lock`. If that fails (typically because the user is running X-windows), it should try to use the `xset` program to change the LED status of *all* LEDs that can be controlled by X (it

depends on installation, and typically includes only the scroll-lock).

- Your programs may make use of the standard commands `ls` and `xset`, executed using one of the `execXX()` functions. Read their manpages to understand how to use them. No other commands are allowed for the execution of your program.
- The `mail_led` program should check once every 10 seconds.
- If `mail_led` found that there is an unexpected error in any steps, it should terminate immediately rather than repeating the error.
- Your programs may make use of `execXX()`, `exit()`, string manipulation functions in the `<string.h>` header file, `printf()`, `scanf()` and their file and string counter-parts `fprintf()`, `fscanf()`, `sprintf()` and `sscanf()`. They may also make use of the macros described in the man pages of `waitpid()` to decode the exit code. No other library routine is allowed. On the other hand, you may use any Linux system calls (i.e., those which man page appears in section 2 in Linux).
- If the `mail_led` program is about to be terminated by `SIGHUP`, `SIGTERM`, `SIGINT` or `SIGQUIT`, it should reset the keyboard LED before terminating. For `SIGQUIT`, you should raise a `SIGQUIT` at the end of the handler rather than exit normally, so as to create a core file as requested by the user.

3. Counting the number of mails in a mailbox

- A mbox format mailbox is a single file holding all mail messages. In our example above, the name of the file is `/var/mail/tmchan`. Each mail message begins with a line that starts with the five characters "From " (note the space after "From"). The mail content is guaranteed to contain no line starting with these 5 characters. (If a sender sends such a mail, a ">" character is added to the beginning of that line.)
- A maildir format mailbox is a directory consisting of a number of files. In our example above, the name of the file is `Mail/new/`. There may be some files for control purposes, which are not of interest for us. We are interested in files with filenames comprising of only digits (a mail in the file with name "123" is the 123-rd mail received). Each of them stores exactly one mail in the mailbox.

4. Useful manpages

- Process and program execution: `fork(2)`, `waitpid(2)`, `exit(3)`, `execve(2)`.
- Handling files: `open(2)`, `close(2)`, `read(2)`, `write(2)`.
- Pipeline: `pipe(2)`, `dup2(2)`.
- Sleeping for sub-second interval: `nanosleep(2)`, `setitimer(2)`.
- Signal handling: `sigaction(2)`.
- Get directory list: `ls(1)`.
- Working with the keyboard LEDs: `ioctl(2)`, `ioctl_list(2)` (look for the "LED" string and read the corresponding include file), `xset(1)`.

5. Handin instructions

Send us your source files `mail_led.c`, `check_mbox.c` and `check_mdir.c` via the handin system of the department. You **must** also handin a `README.txt` file to tell us the group members of your group (even if you are the sole member of the group).

6. Final note

Some of the tasks required by this assignment can be done more easily if we allow you to use library functions like `popen()`, `opendir()` and `readdir()`. Another, perhaps even easier, way is to use a scripting language like Python to perform all the tasks. The assignment chooses a more difficult path, to get you familiar with the use of various system calls.