EECS 3221.3
Assignment 1.

Processes, Signals and Interprocess Communication.

Due Date: Wednesday Oct. 12 at 4:00pm.

1. Exploring Linux Use the command `strace` to trace the system calls of the execution of

```
wc <file>
```

and answer the following questions (in a single plain ascii file):

(1) Why does your process read files like `ld.so*` when it starts up?

(2) Why does it read files like `*locale*` immediately afterwards?

(3) Program `wc` reads the file one chunk at a time. How big is this chunk?

(4) Which system call does not return anything?

2. A Program for Performance Evaluation of Signals

You will write a few programs to estimate the time it takes for two processes to exchange N [default: 100] signals. You will write two versions of it. Each version will consist of two programs: `sigperf1` and `sigshooter1` for the first version and `sigperf2` and `sigshooter2` for the second. In both version the `sigperf` program creates two processes running `sigshooter` and these two send each other `SIGUSR1` signals N times and in the end all three print the CPU and system times. The two children print their own and the parent (`sigperf`) prints the times for the two children. The two children altrnate between sending a signal and waiting for a signal.

The two version differ on how the two children processes (`sigshooter`) the children are informed of each other's process ID.

Version 1

In the first version the parent process creates the two children in succession and the process ID is passed as command line argument. Obviously the process ID of the second child is not known at the time of creation of the first one, so number 0 is given. The second child, that knows the ID of his brother sends him a signal. By setting up the signal handler appropriately the first child can find the IS of his sibling and signal back.

Version 2

In the second version they both discover each other's ID by opening a named pipe (FIFO) each which is provided as argument by their parent. They transmit enough ASCII bytes each to encode the PID. A second argument that can take the values 1 and 2, tells them who is firing first and who is firing second.

3. Programming Standards and Documentation

You have to write high quality code with good error checking and graceful error handling. The return values of all system calls that can return an error have to be checked, all user interaction has to be examined, no buffer overflows should be possible etc. Furthermore no freezing, crashing or deadlocking is acceptable. And synchronization between processes cannot depend on timing.

Each programming assignment version has to be in its own directory named `V1` and `V2` with its own makefile. The makefile allows compilation, cleanup and generation of documentation. There should be at least three targets: `all`, `clean`, and `man`. The first `all` compiles the program, the second removes the executable and all other useless files and fifos, and the third displays the manual. The manual should be plain ascii files and follow the style of man pages in section 1.

4. Submitting the Programming Assignment

The programming assignment has to be submitted online through Moodle. The first question is submitted as q1.txt and the rest as V1.tar and V2.tar.

5. Asking Questions

Please e-mail assignment related questions to minas at cse.... The answer will be posted on the FAQ.

6. Reading

Besides the code in the course web pages, you should read several man pages. Pages like mkfifo(3), kill(2), sigaction(2), fork(2), read(2), write(2), open(2), close(2), signal(7), errno(3) are good starting points.