Lab C: Reading Material

Read this after reading all relevant material from lab 2. Lab C is based on lab 2 and is meant to be done after completing that lab.

Introduction to UNIX/Linux Pipelines

Pipelines are an extremely important concept in system programming. The name "pipeline" itself suggests that pipes are, somehow, involved. But what exactly is a pipeline? This is most easily explained by example, and in our case, by typing "echo spider-pig | head -c 6" in the Linux shell.

The output of this command, as you will notice, is the string spider. However, if you execute "echo spider-pig" and "head -c 6" separately, you see something quite different. The first command simply echoes spider-pig, and the second command asks for user input, a nd prints the first 6 characters. One can easily deduce that when the two commands are chained using the "|" symbol, the output of echo is directly fed into head.

And this is where pipes come into play. The two processes communicate using a pipe which they both share: The standard output of echo is redirected to the pipe's "write-end", and the standard input of head is redirected to the "read-end". A pipeline, then, is simply a chain of processes where the standard output of one process feeds the standard input of the following process. The principle of pipelines easily generalizes to an unlimited number of processes. For instance, the command "echo spider-pig | cat | head -c 6" entails a pipeline consisting of three processes (echo, cat & head) and two pipes.

The History Command

The history command in Linux shells is a built-in shell tool that displays a list of commands used in the terminal session. Configuration of shell history is unique to each shell, e.g. for bash a few parameters in ~/.bashrc determine how ~/.bash_history shall look like (history size, history mode, etc.). A few examples relevant to this lab are shown below.

List last 5 commands:
```
#>history 5
315 16:04 touch a
316 16:04 date
317 16:04 rm -f a
318 16:04 pwd
319 16:04 history 5
```

Issue a new command: `#>echo "ONE"`
```
ONE
```

Repeat the last issued command using !!
```
#>!!
```

```
echo "ONE"
ONE
```

List last 5 commands (note: !! isn't included in the list)

```
#>history 5
318 16:04 pwd
319 16:04 history 5
320 16:05 echo "ONE"
321 16:05 echo "ONE"
322 16:05 history 5
```

Re-do command 318: pwd (note: !n itself will also be excluded from history list)

```
#>!318
pwd
/home/users/phd/zakhr/
#>
```

MAN: Relevant Functions and Utilities

close(2), open(2), dup(2), dup2(2), pipe(2), fork(2), execvp(3)