Pipes (in Ch 15)

Interprocess communication using Pipes

- Shell's "view"   cmd1 | cmd2 | cmd3 | ... | cmdn
  - shell's stdin attached to cmd1 stdin
  - cmd1 stdout attached to cmd2 stdin
  - cmd2 stdout attached to cmd3 stdin
  - ...
  - cmdn-1 stdout attached to cmdn stdin
  - cmdn stdout attached to shell's stdout

View:

- Similar to a water pipe

- Filter -- sort
  - ps aux | grep xfce | sort
  - who | cut -c1-8 | sort | uniq
  - grep "(int" *.c | cut -f1 -d: | tee LIST | sort | uniq

System call

```
int pipe ( int filedes[2] );

    int fd[2];

    if (pipe (fd) < 0)  sys_err("pipe");
```

- fd[0] -- read end of the pipe
- fd[1] -- write end of the pipe

Termination:

- Close pipe end.
- read on fd[0] when all fd[1] ends are closed => EOF
- write on fd[1] when all fd[0] ends are closed => error and SIGNAL (SIGPIPE)

Parent -> Child

```
int fd[2];  int pid;

if (pipe (fd) < 0) sys_err("pipe");
if ( (pid = fork()) < 0) sys_err("fork");
if (pid > 0) {
   close (fd[0]);
   ---- write on fd[1]);
   close (fd[1]);  /* Ends communication */
} else {
   close (fd[1]);
   ---- read on fd[0];
   close (fd[0]);  /* After EOF found on fd[0]. */
}
```

(No execve here!)

Child -> Parent

```
int fd[2];  int pid;

if (pipe (fd) < 0) sys_err("pipe");
if ( (pid = fork()) < 0) sys_err("fork");
if (pid > 0) {
   close (fd[1]);
   ---- read on fd[0];
   close (fd[0]);  /* After EOF found on fd[0]. */
} else {
   close (fd[0]);
   ---- write on fd[1]);
   close (fd[1]);  /* Ends communication */
}
```

popen / pclose (Stdio)

- FILE *popen(const char *command, const char *type);
  - Type => r, w, r+
  - Runs command with /bin/sh

- int pclose(FILE *stream);
  - Waits & returns exit status (wait4 value)

Another use of pipes: command expansion

bash:

- N=$(expr $N + 5)

ush:

- envset N $(expr ${N} + 5)

cmd1 $(cmd2 ...)

- output is piped back to shell for cmd2
- output from cmd2 becomes part of command line for cmd1

How to get it done?

```
pipe(fd); // Create the pipe
fork();   // Create the new process
in child:
    dup2(fd[1],1)
    close(fd[0])
    close(fd[1])
    execxx(cmd2...)
in parent
    close(fd[1])
    read fd[0] until EOF
    close(fd[0])
```

Time for a4 slides.