Features/Issues

- Pipeline in foreground
- Pipeline in background
- Stdin/stdout redirection
- Handling ctrl-c from keyboard
  » Should kill all processes in a foreground pipeline
    * `ls -l | sort -n +4 | more`
  » What about background pipelines???
- Background process termination notice
- Miscellaneous
  » Parsing pipeline
  » Check #args to built-in commands

Web Resources

- test40.C
  » Shows effect of handling or ignoring SIGINT (ctrl-c)
- doio.c
  » Shows: read stdin/write stdout using unbuffered I/O
- Project A command parser
  » Can be used to parse one command (no pipe)
  » But uses STL ‘map’
- Homework 5, Problem 6 solution (To Come)
  » Shows how to separate plumbing and I/O
    * But different structure than Project B
  » `do_plumbing(); do_io();`
    * `do_plumbing: for (int i=0; i<npipes; i++) { Pipe(fd); ... }`
    * `do_io: while (Read(STDIN_FILENO, ...) > 0) Write(STDOUT_FILENO, ...);`

Pipeline (Review)

- Example (sort by non-decreasing file size)
  » `ls -l | sort -n +4`

Web Resources

- test40.C
  » Shows effect of handling or ignoring SIGINT (ctrl-c)
- doio.c
  » Shows: read stdin/write stdout using unbuffered I/O
- Project A command parser
  » Can be used to parse one command (no pipe)
  » But uses STL ‘map’
- Homework 5, Problem 6 solution (To Come)
  » Shows how to separate plumbing and I/O
    * But different structure than Project B
  » `do_plumbing(); do_io();`
    * `do_plumbing: for (int i=0; i<npipes; i++) { Pipe(fd); ... }`
    * `do_io: while (Read(STDIN_FILENO, ...) > 0) Write(STDOUT_FILENO, ...);`

Web Resources

- test40.C
  » Shows effect of handling or ignoring SIGINT (ctrl-c)
- doio.c
  » Shows: read stdin/write stdout using unbuffered I/O
- Project A command parser
  » Can be used to parse one command (no pipe)
  » But uses STL ‘map’
- Homework 5, Problem 6 solution (To Come)
  » Shows how to separate plumbing and I/O
    * But different structure than Project B
  » `do_plumbing(); do_io();`
    * `do_plumbing: for (int i=0; i<npipes; i++) { Pipe(fd); ... }`
    * `do_io: while (Read(STDIN_FILENO, ...) > 0) Write(STDOUT_FILENO, ...);`
Pipeline Movie (1)

- **xssh child:**
  - [0] stdin
  - [1] stdout
  - [2] stderr

- **xssh child: pipe**
  - [0] stdin
  - [1] stdout
  - [2] stderr
  - fd[0] = [3] pipe.I

- **xssh child: fork**
  - [0] stdin
  - [1] stdout
  - [2] stderr

Pipeline Movie (2)

- **xssh child:**
  - [0] stdin
  - [1] stdout
  - [2] stderr

- **child:**
  - dup2(fd[1], X);
  - close(fd[0]);
  - close(fd[1]);

- pipe.I
  - pipe.O

  - child

Very Important:
- Close both ends of all pipes

Aside:
- If xssh creates more than one pipe, close both ends of ALL pipes before using them

Foreground Pipeline

- **Parsing**
  - "ls -l /usr/bin | sort -n +4 | more" becomes
  - i.e., 3 strings, each representing a command
  - We know how to handle individual commands

- **Approach**
  - xssh sets up pipes for 3 children and fork-execs the 3 processes for 'ls', 'sort' and 'more'
  - Creates 4 processes in the same process group
- **Ctrl-c:** SIGINT sent to and terminates all 4 processes !!!

- **Need To:**
  - Put the 3 children in a new process group
  - xssh should catch SIGINT; send SIGINT to processes

Process Group Example

- **script0**
  - #!/bin/sh
  - echo "pid of this shell is $$"
  - ls -l /usr/bin | sort -n +4 | mysleep $1 &
  - ps -o pid,ppid,pgrp,cmd

- **script0 Output from "script0 60"**
  - pid of this shell is 27129
  - 17233 17232 17233 -tcsh
  - 27129 27129 27129 /bin/sh ./script0 60
  - 27131 27129 27129 sort -n +4
  - 27132 27129 27129 /bin/sh ./mysleep 60
  - 27133 27129 27129 ps -o pid,ppid,pgrp,cmd
  - 27134 27132 27129 sleep 60

- **"kill -9 -27129" kills all processes in group 27129**
Process Group

- **Def.** A collection of processes that can receive signals from the same terminal
- Every process belongs to some process group
- A *process group leader* is a process whose PID is the same as its PGRP
  - The leader can die or join another group
- A child created by `fork()` inherits the PGRP of the parent

Some functions

```c
int getpgrp(void)
```
- Return PGRP of caller

```c
int setpgid(pid_t pid, pid_t pgid)
```
- Change PGRP of child process `pid` (current process if 0)

Beware of race conditions!!!

xssh

- Is in same process group as the shell that exec'd it
- Should ignore SIGINT
  - `ctrl-c` will terminate it's foreground children but not xssh
  - Won't terminate xssh's parent (login shell) either
    - Shell's ignore SIGINT
- Change process group of background pipelines
  - `ctrl-c` will have no effect on these processes