CSE410 Project 2
Spring 2018

02/02/2018

- 100 points + 10 (E.C.)
- Deadline
  - **Thursday, Feb. 22, 2018 at 11:59 pm**
  - Handin
    - http://secure.cse.msu.edu/handin/
- Compile & Run on “cse410”

Tasks

- Assignment Notes #6
  - MyShell::ExecuteCShellCommand()
  - MyShell::PrintEnv()
  - MyShell::Prompt()
  - MyShell::PrintPromptInfo()
  - MyShell::PrintHistory()
  - MyShell::UpdateHistory()
  - MyShell::GetUserName()
  - MyShell::GetHostName()
  - MyShell::GetTime()
  - MyShell::GetCurrentDirectory()
  - MyShell::ChangeCurrentDirectory()
  - MyShellProcess::ExecuteInBackground()
  - MyShellProcess::Execute()

execv()

- Execute a new program
  - `./myShell -c ls -l`
  - `<1 host:userName> ls -l`
- How to use it?
  - `execv(myAgrs[0], myArgs);`
    - Never returns on success
    - Returns -1 on error, always check for error condition and respond appropriately should it occur.
  - myArgs (an array of char* with size 3)
    - myArgs[0]
      - "/bin/ls"
    - myArgs[1]
      - "-l"
    - myArgs[2]
      - NULL

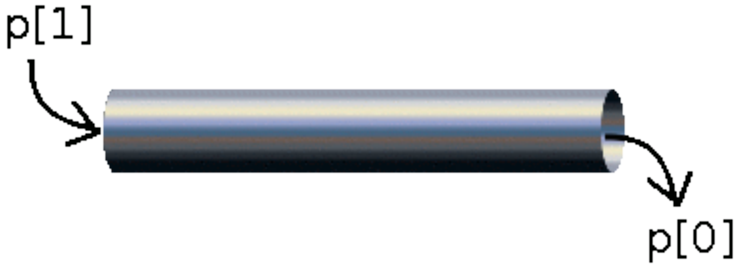
fork()

- Create a new process
  - The child process is an almost exact duplicate of the parent process
  - Each process modifies the variables in its own stack, data, and heap segments without affecting the other
- Values returned from `fork()`
  - -1, fork() failed
  - 0, in child process
  - >0, in parent process (process id of the child)

```
switch(fork())
{
      case -1:
            // fork() failed
            break;
      case 0:
            // we are in child process
            break;
      default:
            // we are in parent process
            break;
}
```

pipe()

- Allow the output produced by one process to be used as the input of the other process
  - In CShell
    - `ls -la | grep "fileName"`
    - The shell creates **2** processes executing "ls" and "grep" respectively
  - In myShell?
    - pipe()
    - fork()
    - execv()



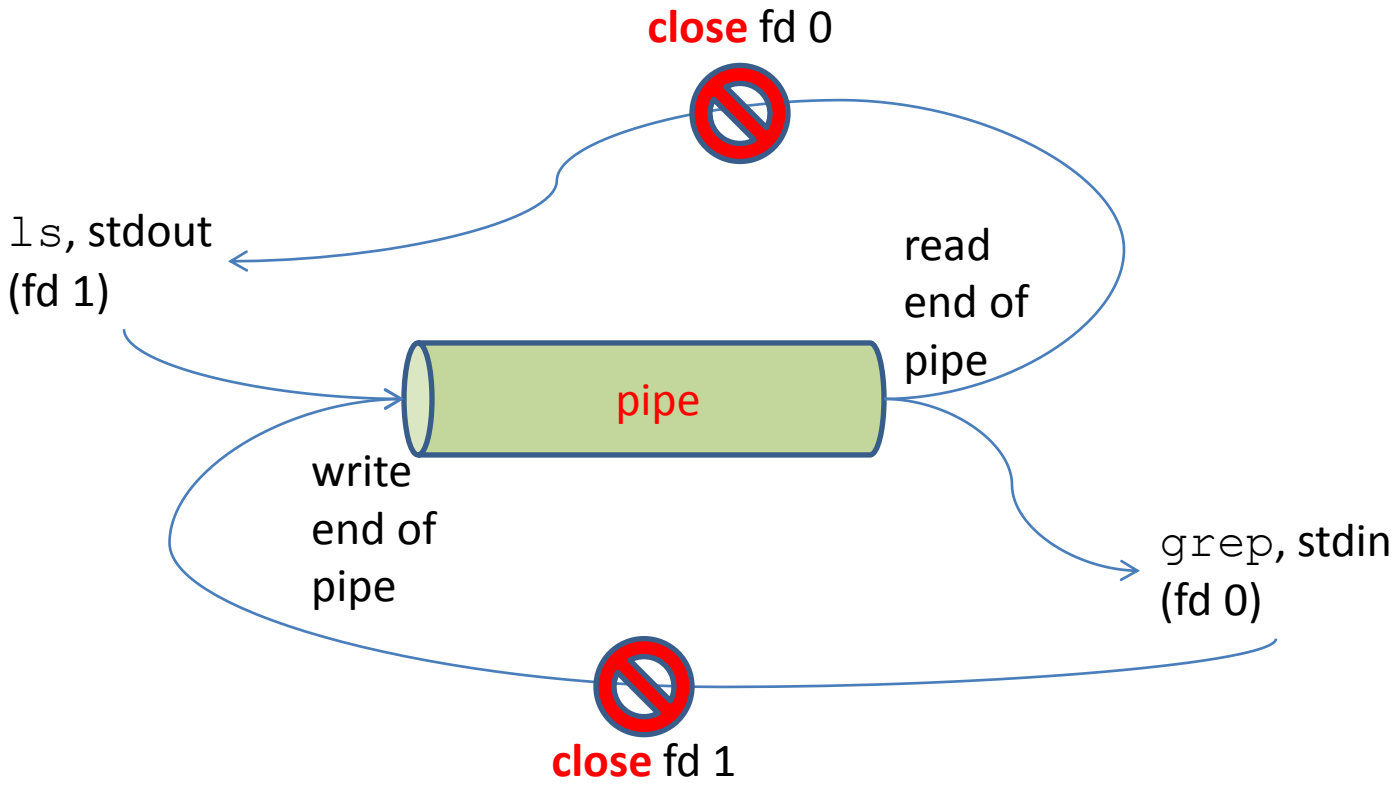
```
int fd[2];
if(pipe(fd)==-1)
{
     // pipe error
}

switch(fork())
{
     case -1: // fork error
                  break;

     case  0: close(fd[1]);
              // child reads from pipe
              break;

     default: close(fd[0]);
              // parent writes to pipe
              break;
}
**Note:  close() returns -1 on errors
```

How does the pipe look like?



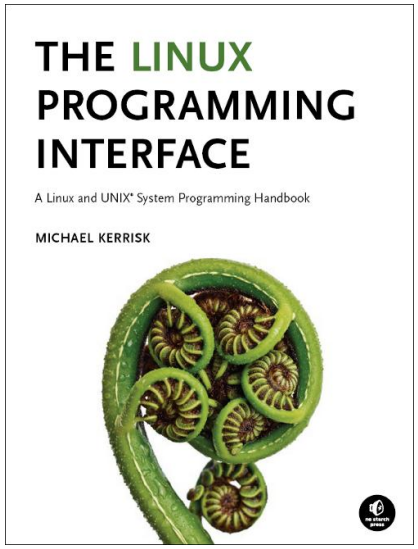
Hints

- Bind stdout/stdin to the pipe
  - dup2()
  - STDIN_FILENO or STDOUT_FILENO
- IO Redirection
  - “>”, output redirection
  - “<“, input redirection
  - open()
  - dup2()
  - STDIN_FILENO or STDOUT_FILENO
- Don’t forget to close the unused file descriptors

Deliverables

- **yourNetID**.zip
  - Project2.cpp
  - *.cpp
  - *.h
  - Makefile
  - Readme (optional)
- Handin system
- Compile & Run on “cse410”

References



- System calls
  - http://linux.die.net/man/3/exec
  - http://linux.die.net/man/3/fork
  - http://linux.die.net/man/3/pipe
  - http://linux.die.net/man/3/dup2
  - http://linux.die.net/man/3/open
- Figures
  - http://en.wikipedia.org/wiki/Shell_Oil_Company
  - http://www.yourdictionary.com/fork
  - http://www.reloco.com.ar/linux/prog/pipes.html
  - http://nostarch.com/tlpi