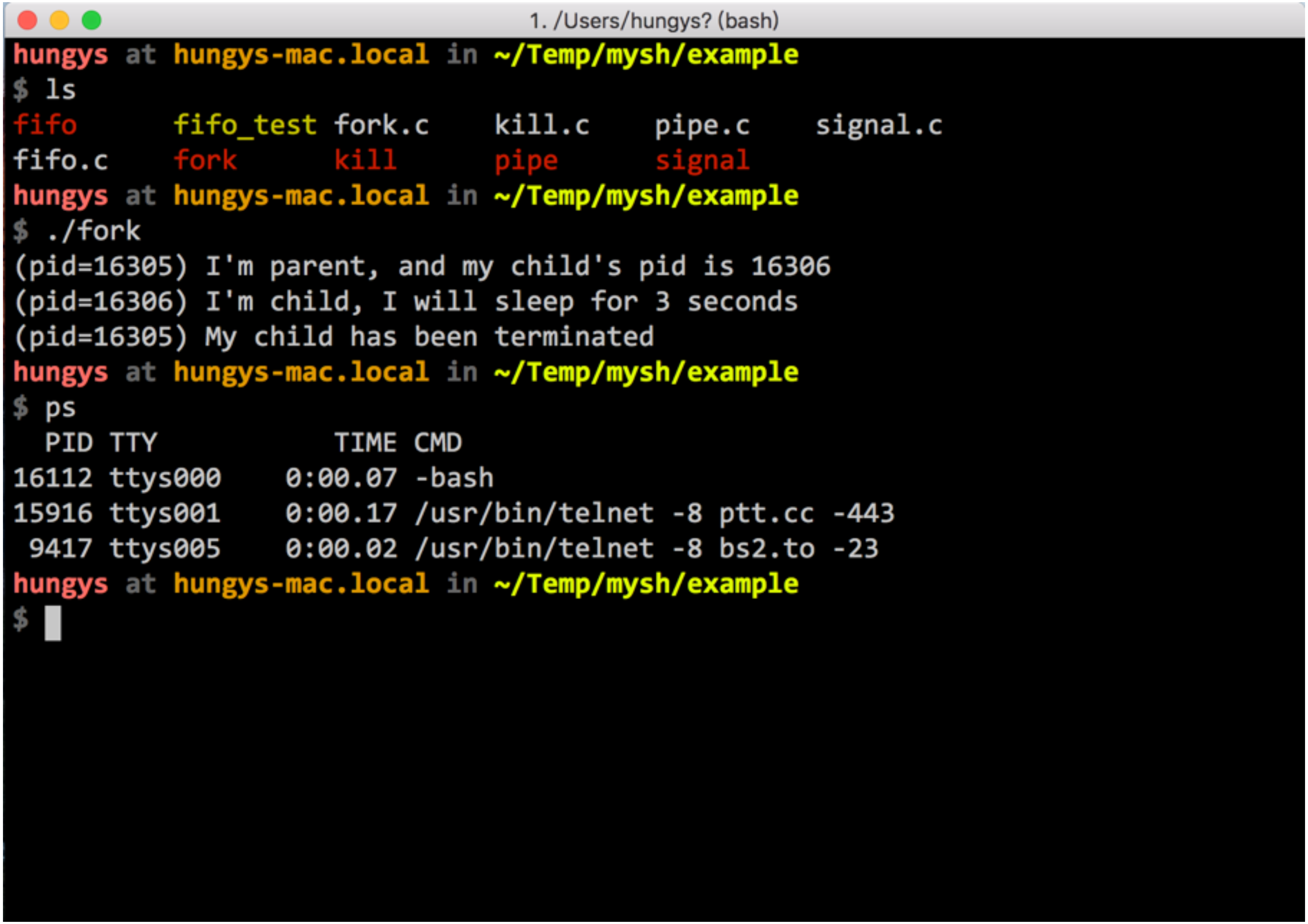
HW1: Process & IPC

walkthrough by TA

Shell?



```
hungys at hungys-mac.local in ~/Temp/mysh/example
$ ls
fifo       fifo_test fork.c    kill.c    pipe.c    signal.c
fifo.c     fork      kill      pipe      signal
hungys at hungys-mac.local in ~/Temp/mysh/example
$ ./fork
(pid=16305) I'm parent, and my child's pid is 16306
(pid=16306) I'm child, I will sleep for 3 seconds
(pid=16305) My child has been terminated
hungys at hungys-mac.local in ~/Temp/mysh/example
$ ps
  PID TTY           TIME CMD
16112 ttys000    0:00.07 -bash
15916 ttys001    0:00.17 /usr/bin/telnet -8 ptt.cc -443
 9417 ttys005    0:00.02 /usr/bin/telnet -8 bs2.to -23
hungys at hungys-mac.local in ~/Temp/mysh/example
$ 
```

Write your own!

mysh (=my shell)

- Execute programs in foreground or background (&)
- Basic job control functions (i.e. bg, fg)
- Support pipeline (e.g. progA | progB | progC)

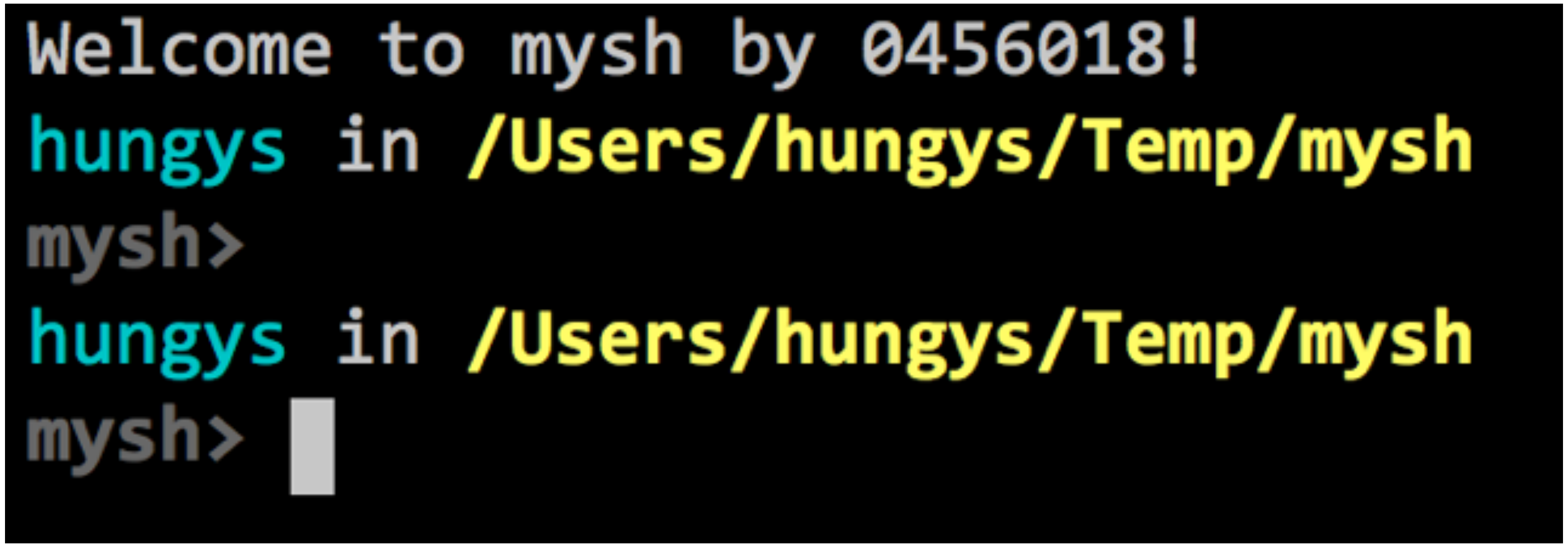
Environment

- Demo on NCTU CSCC workstation (i.e. **linux1~6**)
- or Ubuntu 14.04.3 LTS
- **C/C++** only!

Tasks

- 9 tasks
- no starter code, but many **hints** are provided
- Possible to done within 400 lines of code!

#1 Shell Prompt



```
Welcome to mysh by 0456018!
hungys in /Users/hungys/Temp/mysh
mysh>
hungys in /Users/hungys/Temp/mysh
mysh> 
```

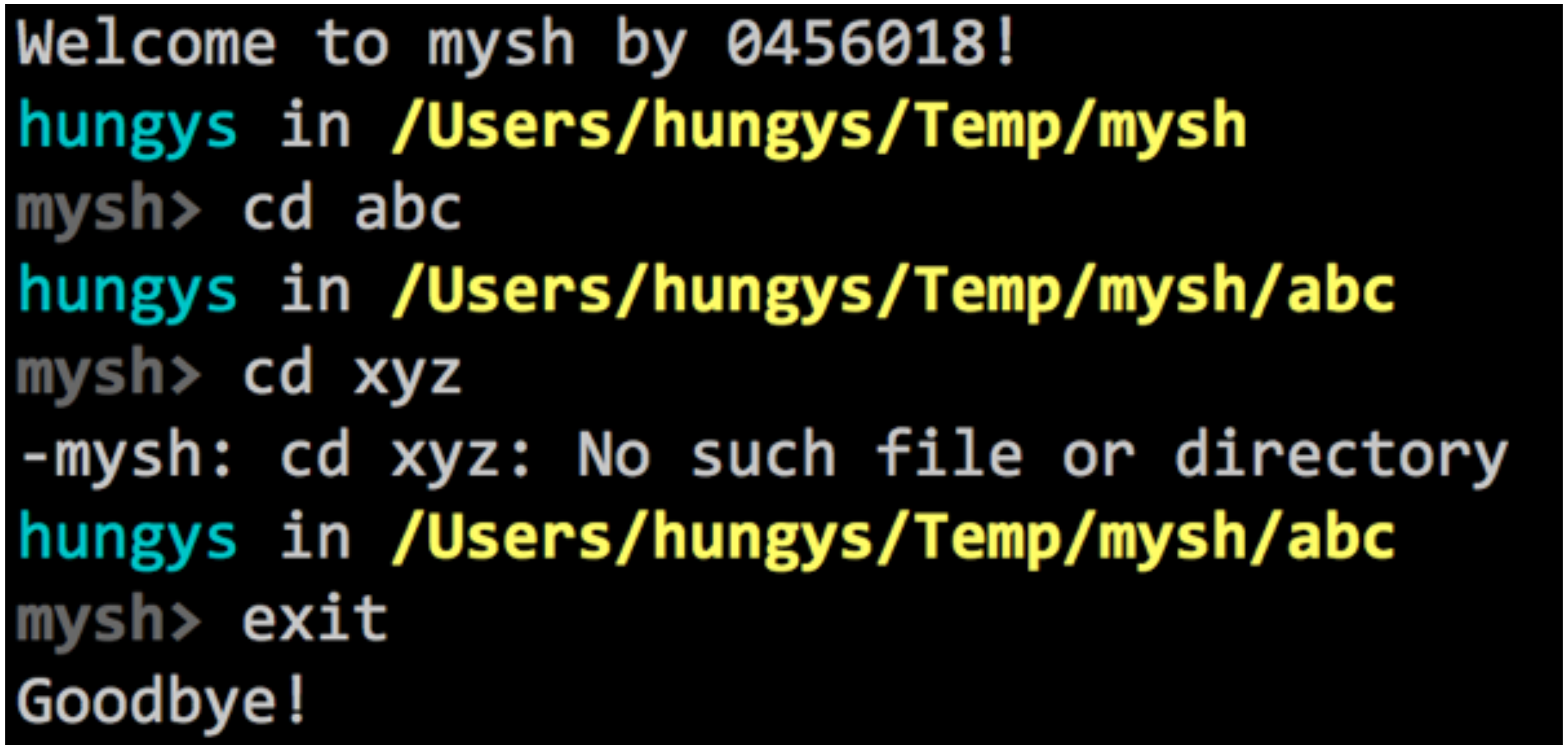
[Hint] getpwd(), getlogin_r()

#2 Command Parser

- **progA [argA1 argA2 ... argAN] l progB [argB1 argB2 ... argBN] l ... l progZ [argZ1 argZ2 ... argZN] [&]**
- flex and yacc/bison are allowed, but not necessary
- Example
    - > prog
    - > prog &
    - > prog 10 20
    - > progA 10 | progB | progC
    - >   progA| progB| progC   &

#3 Internal Commands

- exit
- cd <path>



```
Welcome to mysh by 0456018!
hungys in /Users/hungys/Temp/mysh
mysh> cd abc
hungys in /Users/hungys/Temp/mysh/abc
mysh> cd xyz
-mysh: cd xyz: No such file or directory
hungys in /Users/hungys/Temp/mysh/abc
mysh> exit
Goodbye!
```

[Hint] chdir()

#4 Program Execution

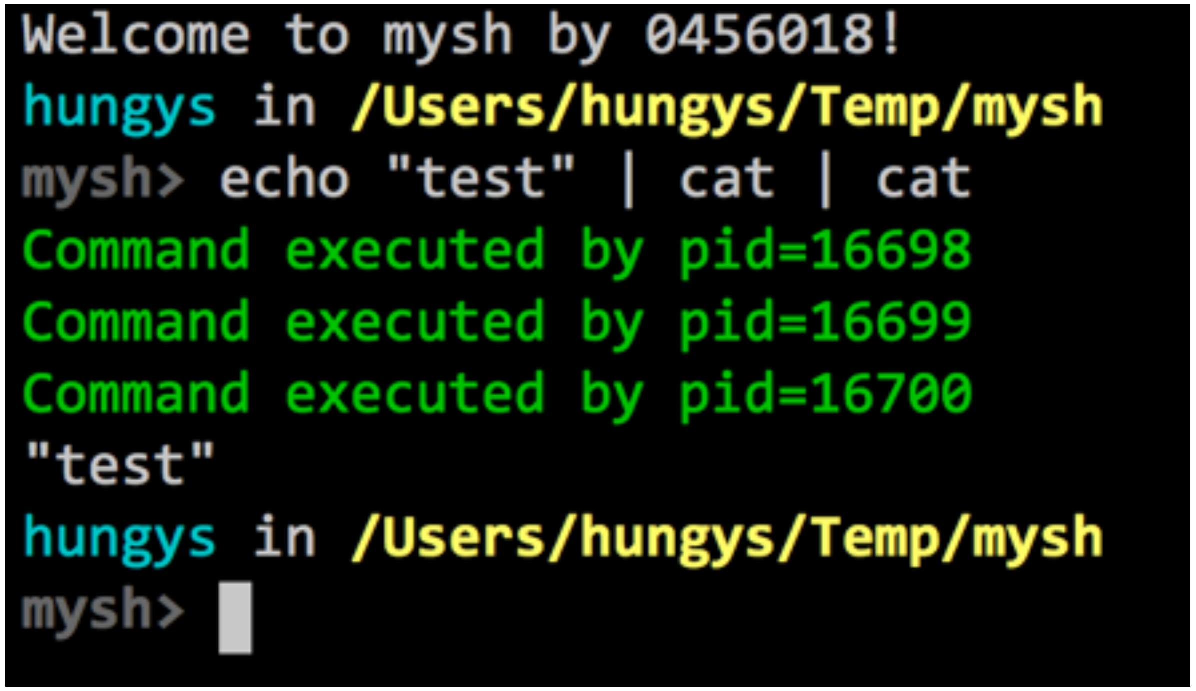
- Foreground execution
- Background execution (e.g. ./prog **&**)
- Print “Command executed by pid=<pid> [in background]” before execution
- **DO NOT use system()!**

[Hint] fork(), execvp(), waitpid()

#5 Shell Pipeline

- progA argA1 argA2 | progB argB1

- Fork both progA & progB, make the output of progA to the input of progB



```
Welcome to mysh by 0456018!
hungys in /Users/hungys/Temp/mysh
mysh> echo "test" | cat | cat
Command executed by pid=16698
Command executed by pid=16699
Command executed by pid=16700
"test"
hungys in /Users/hungys/Temp/mysh
mysh> 
```

[Hint] fork(), execvp(), waitpid()

#6 Signal Handling

- **Control-C** generates SIGINT
  - Kill foreground job
- **Control-Z** generates SIGSTOP
  - Suspend foreground job

[Hint] signal(), sigaction(), SIG_DFL, SIG_IGN

#7 Job Control

- **fg <pid>**: bring process to foreground
- **bg <pid>**: bring suspended process to background
- **kill <pid>**: terminate process/process group
- Process group: how to kill progA | progB | progC?

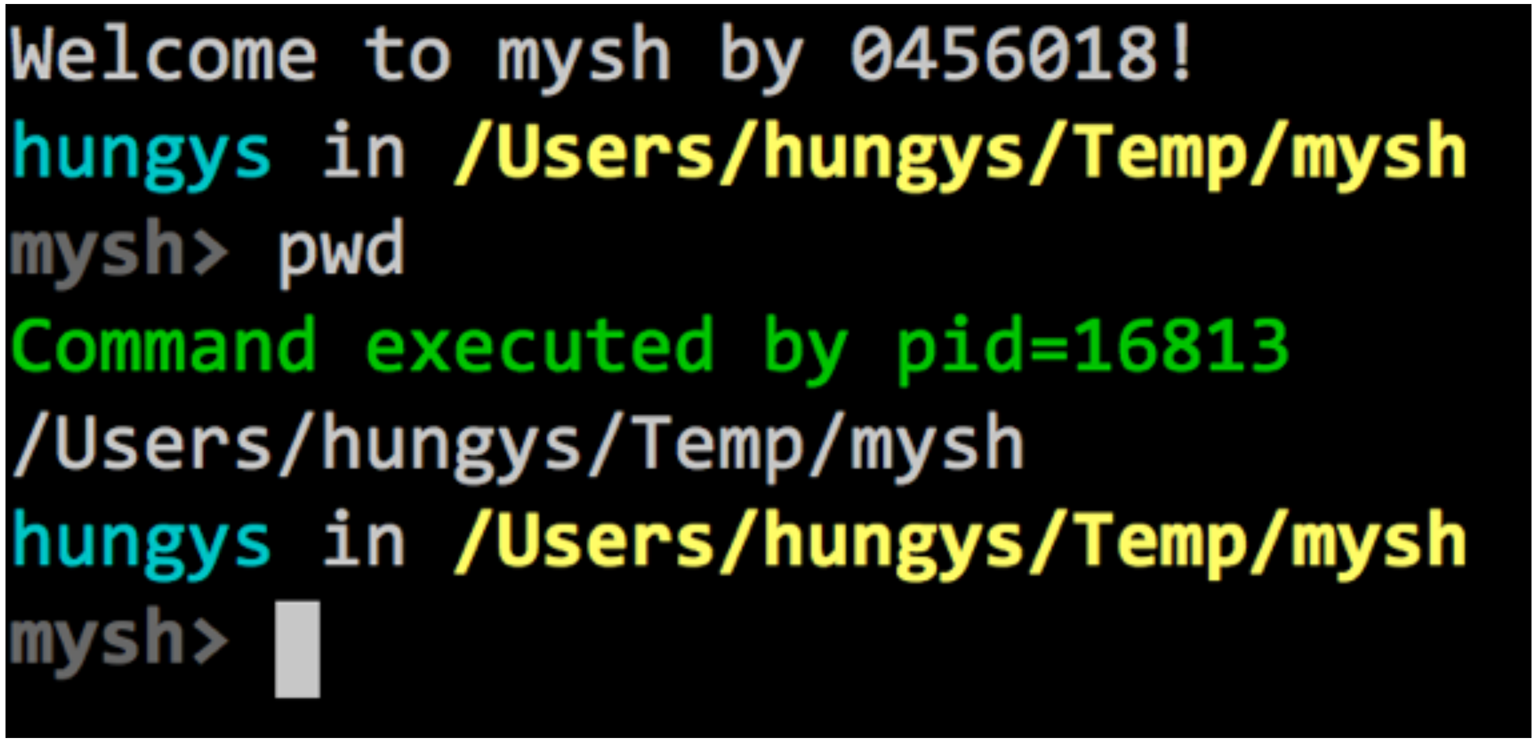
[Hint] setpgid(), kill(), tcsetpgrp(), SIGCONT

#8 Prevent Zombies

- Exit status not read via the **wait()** sys call
- May cause resource leak
- Consider a background job

[Hint] waitpid(), SIGCHLD

#9 Colorizing Your Shell



```
Welcome to mysh by 0456018!
hungys in /Users/hungys/Temp/mysh
mysh> pwd
Command executed by pid=16813
/Users/hungys/Temp/mysh
hungys in /Users/hungys/Temp/mysh
mysh>
```

[Hint] Add something to what you print

Grading Policy

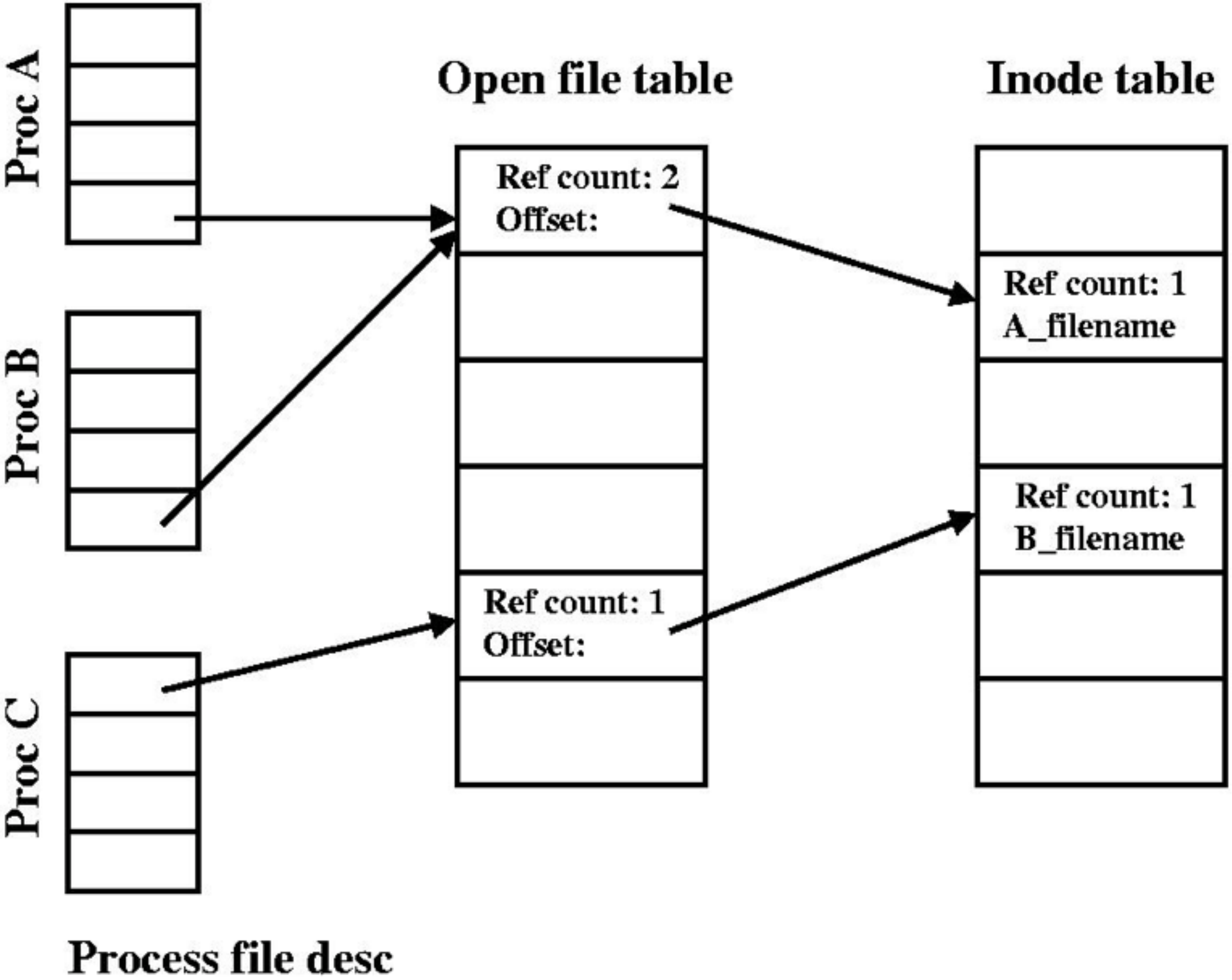
- Part A - 0%
- Part B - **total=90%, max=80%**
  - See details in spec
  - **No need to write a report**
- Part C - 20%

2015/10/31 23:59:59

Any Question?

- TA: 洪聿昕 (Yu-Hsin Hung)
- E-mail: hungys.cs04g@nctu.edu.tw
- Subject: **[OS] HW1 Question (<STUDENT ID>)**

Appendix



Proc A

Proc B

Proc C

Process file desc

Open file table

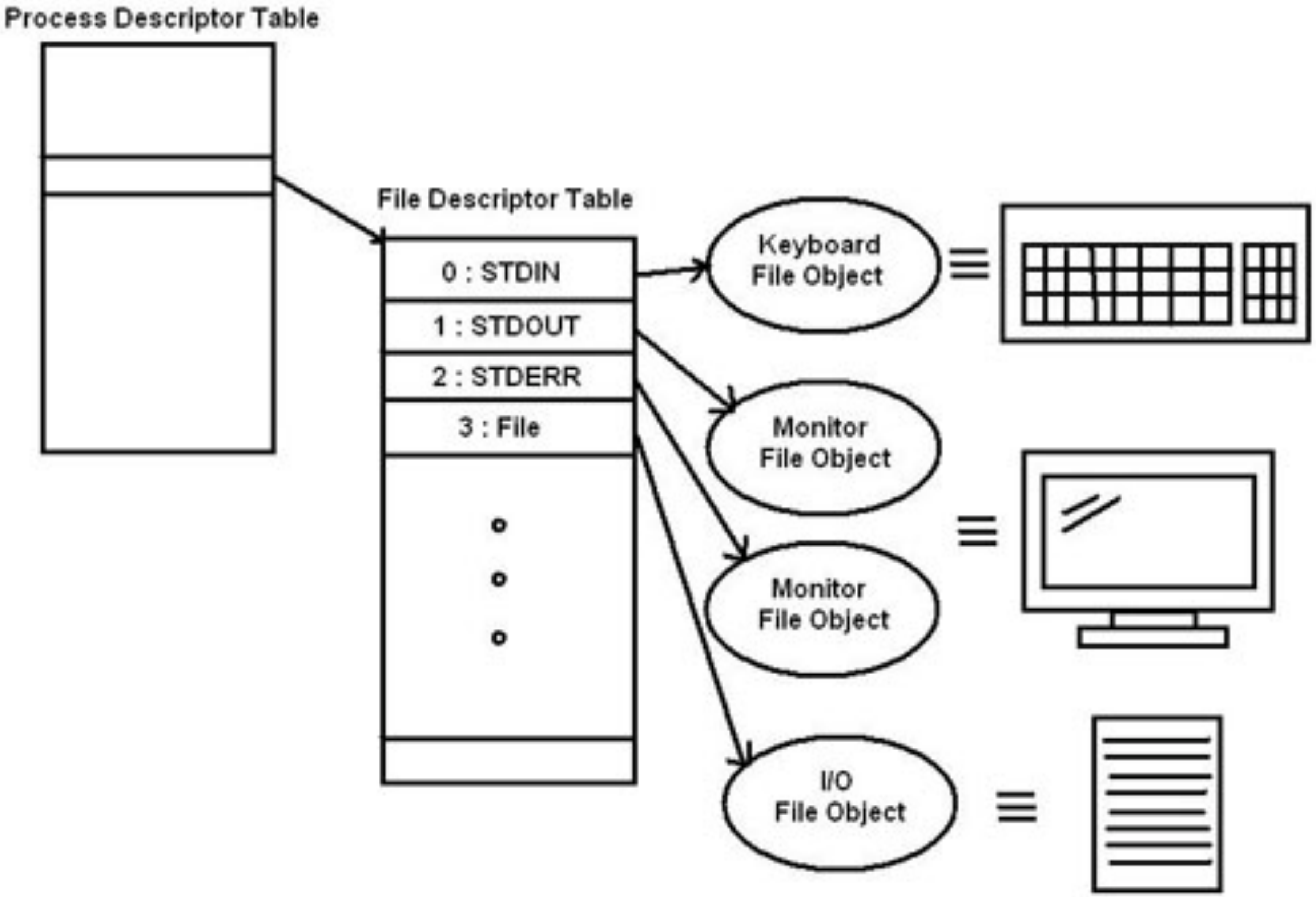
Ref count: 2
Offset:

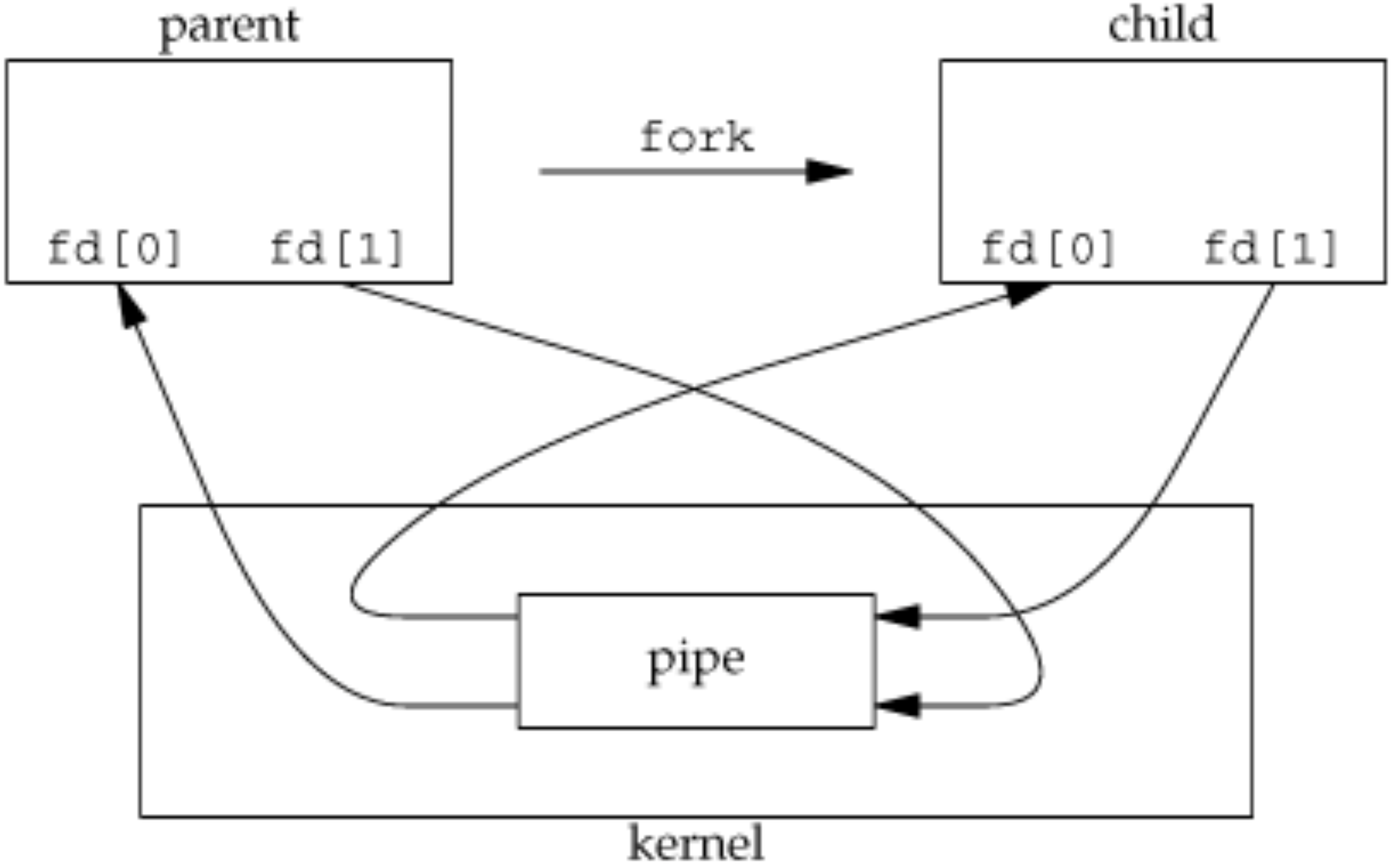
Ref count: 1
Offset:

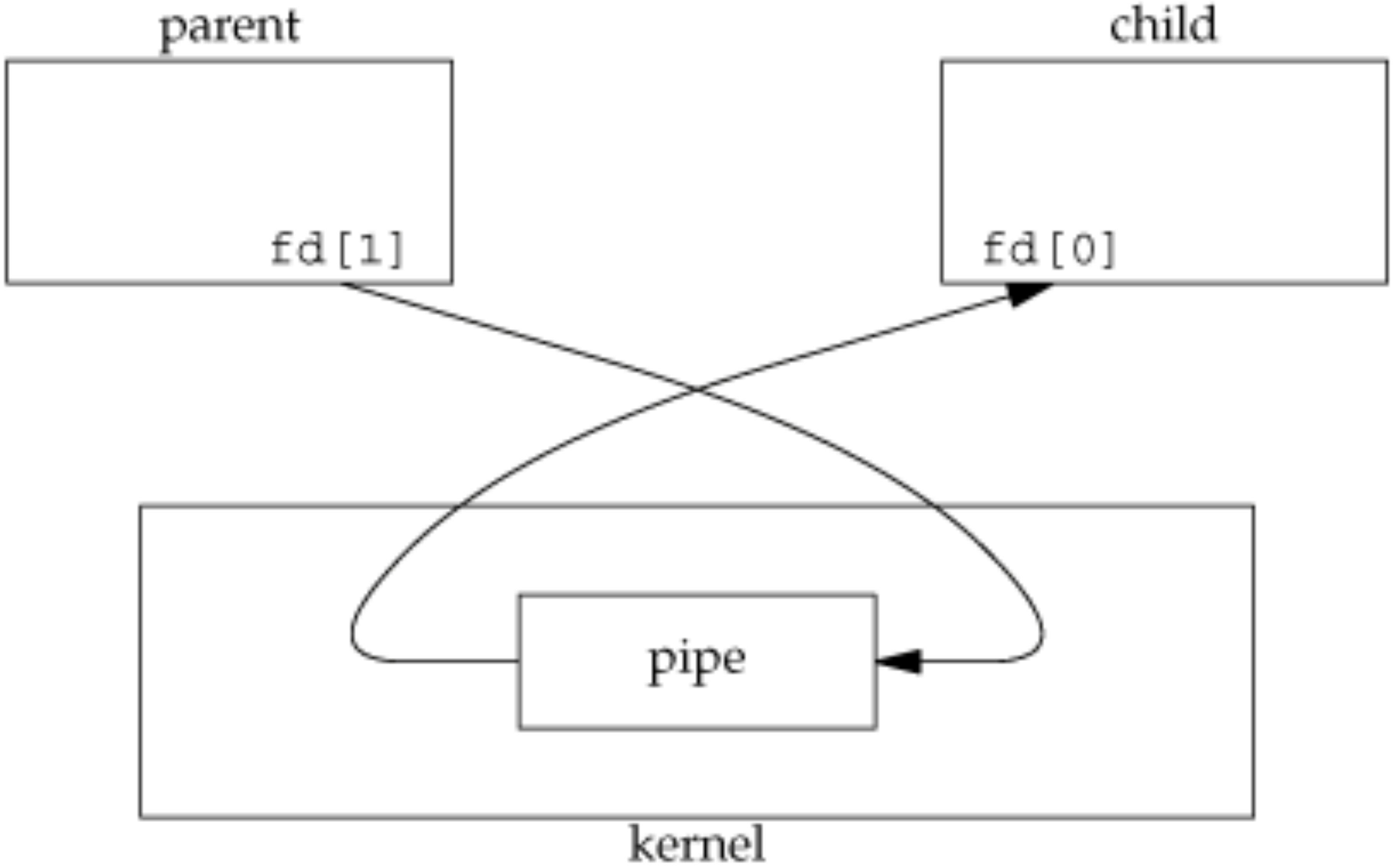
Inode table

Ref count: 1
A_filename

Ref count: 1
B_filename


Process Descriptor Table
File Descriptor Table
0 : STDIN
1 : STDOUT
2 : STDERR
3 : File
Keyboard File Object
Monitor File Object
Monitor File Object
I/O File Object
≡
≡
≡


parent
child
fork
fd[0]
fd[1]
fd[0]
fd[1]
pipe
kernel


parent
child
fd[1]
fd[0]
pipe
kernel