TA. YeouGyu Jeong
(81887821@snu.ac.kr)
System Software & Architecture Lab.
Seoul National University

Spring 2020

4190.307: Operating Systems Lab. 2

Reminder) Late submission policy

- You can use up to 5 *slip* days for this semester
  - You should explicitly declare the number of slip days to use in the Q&A board on the submission server
  - https://sys.snu.ac.kr/main.php?classIdx=1&menu=Board
- 25% penalty per day after *slip* day

Project #3 – Terminating processes

- In this project, you have to
  - Extend the `kill` system call
  - Form process groups
  - Make ctrl-c work
- Due date is April 12(Sunday)

Extending `kill` system call

- `int kill(int pid)`
- If **`pid`** is positive, specified process is killed (current behavior)
- If **`pid`** is negative, processes whose pgid is **`-pid`** must be killed
- If **`pid`** is 0, processes in the calling process's process group must be killed

- On success (at least one process was terminated), return 0
- On error, return -1

Forming process groups

- You have to maintain the process group from `sh` program in the user space using **`setpgid`** system call
- All processes from the same command line should have the same process group id

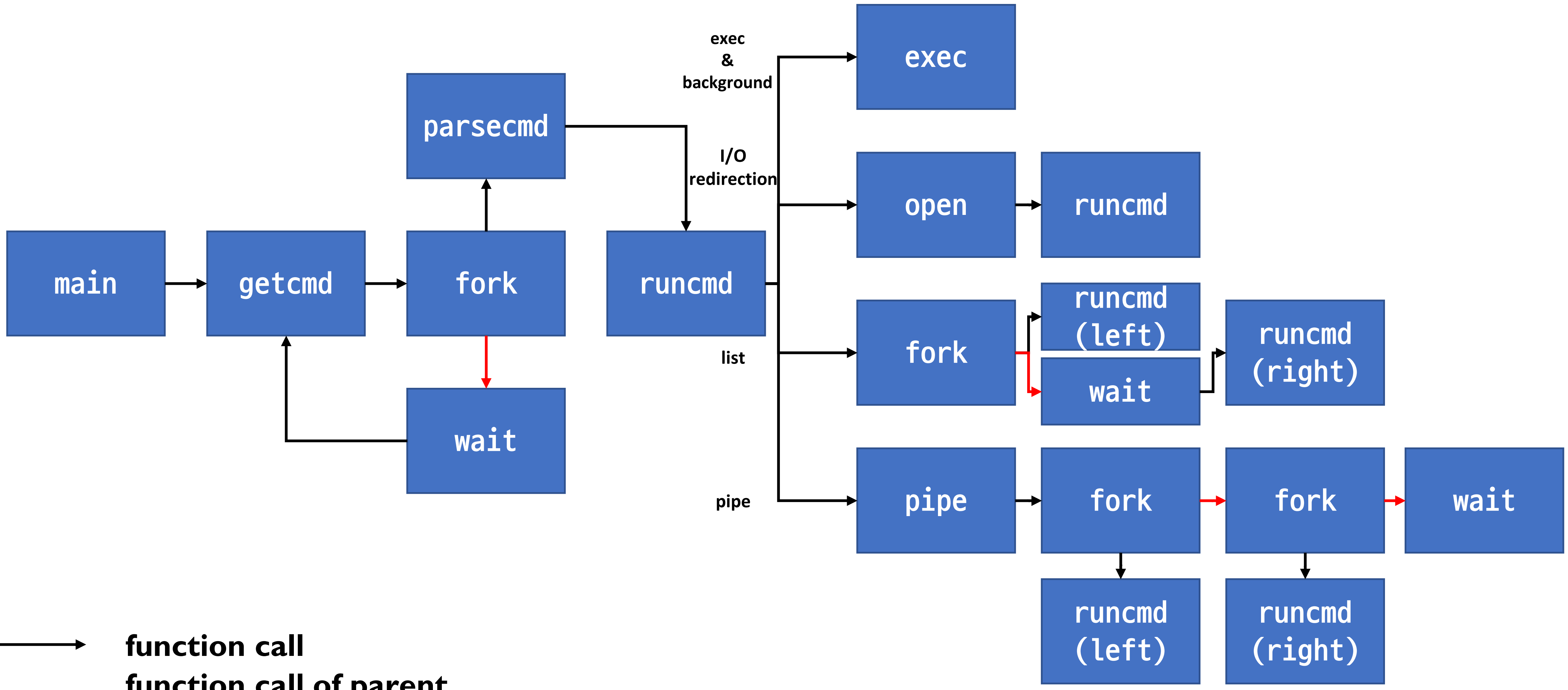
```
$ cat | cat | cat
```

```
1 1 sleep init
2 1 sleep sh
3 3 sleep sh
4 3 sleep cat
5 3 sleep sh
6 3 sleep cat
7 3 sleep cat
```

xv6 sh command types

- EXEC: Execute command
  - e.g.) command
- REDIR: I/O Redirection
  - e.g.) command > file
- LIST: Sequential command execution
  - e.g.) left; right
    right is executed after left is done
- PIPE: Pipelining
  - e.g.) left | right
- BACK: Execute in background
  - e.g.) command &

Understanding xv6 sh



function call
function call of parent (after fork)

Making ctrl-c work

- When ctrl-c is pressed, foreground process group should be killed

```
$ cat | cat | cat
```

```
1 1 sleep init
2 1 sleep sh
3 3 sleep sh
4 3 sleep cat
5 3 sleep sh
6 3 sleep cat
7 3 sleep cat
```

```
$ cat | cat | cat
^c
```

```
1 1 sleep init
2 1 sleep sh
```

Determining foreground process group

- When ctrl-c is pressed, the kernel handles the interrupt
- However, sh in user space knows the foreground process group id
- How can we make our kernel know what foreground pgid is?

How does linux manage foreground processes?

- Linux kernel uses `struct tty_struct` to manage information about terminals
  - It contains foreground process group id(member variable `pgrp`) and a lot of other data
- And the shell is responsible for maintaining foreground pgid via `tcsetpgrp` system call

Starting your project

- You have to start the project from your previous project code
- We will provide you additional user space program for your project
  - `infloop`: Runs infinite loop
  - `fork10`: Creates 10 children processes and runs infinite loop
- To get the codes,
  - First, commit your works using `git commit`
  - Then, merge changes from upstream using `git pull`
  - If you are using branch other than pa2, do `git fetch` and `git merge origin/pa2`

Resolving merge conflict

- `git status` to find out which files were conflicted

```
Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   Makefile
```

- Find out the conflicts and fix them

```
<<<<<<< HEAD
PANUM = PA2
STUDENTID = 2020-00000
=======
PANUM = PA3
STUDENTID =
>>>>>>> 52ba5b10726c1a12b2af4e09a455fbadaef22efa
```

What you have modified

Modification from upstream

```
PANUM = PA3
STUDENTID = 2020-00000
```

- `git add` the files you fixed and finish merging by `git commit`

When you do your project,

- Please only modify Makefile, files in kernel directory, and user/sh.c
  - Changes to other source will be ignored by grading script
- Please remove all the debugging outputs before you submit
- Also, please read the project description carefully
  - https://github.com/snu-csl/os-pa3

You may want to see…

- kernel/defs.h
  - For function definitions
- kernel/proc.h, kernel/proc.c
  - For extending kill system call and process group management
- kernel/console.c
  - For console interrupt handling
- user/sh.c
  - For forming process groups

Thank you!

- Any questions?
- Or feel free to ask us in KakaoTalk