Project2: rwg System

NP TA 垣佑

11/10 18:20

Project 2 Deadline

Demo: 11/12 Thu.

rwg - Remote Working Ground

- Chat-like system
- Provide all functions in **project 1**
- New functions
  - **User pipe**
  - **who - get information of all users**
  - **name - rename**
  - **tell - send message to someone**
  - **yell - broadcast message**

2 Servers + 1 Bonus

- np_simple (Single user)
  - **Project 1**
  - **Concurrent connection-oriented**
- np_single_proc (Multiple users)
  - **Project 1** + **User pipe** + **4 functions** + **Broadcast message**
  - **Single-process concurrent**
- **(Bonus)** np_multi_proc (Multiple users)
  - **Project 1** + **User pipe** + **4 functions** + **Broadcast message**
  - **Concurrent connection-oriented** + **FIFO** + **Shared memory**

Project 2: Submission

- Create a directory named as your student ID, put all files into the directory.
- You must provide Makefile. Two executable files named **np_simple** (server 1) and **np_single_proc** (server 2) should be produced after typing make command.
- You are **NOT** allow to demo if we are unable to compile your project with a single make command.
- Upload only your code and Makefile. DO NOT upload anything else (e.g. noop, removetag, test.html, **.git**, **__MACOSX**)
- zip the directory and upload the .zip file to new e3 platform

ATTENTION! We only accept .zip format

Project 2: About Bonus

- Submission
  - Same rules as previous slide
  - The executable file **np_multi_proc** (server 3) should be produced after typing make command
  - Submit to “**Project Bonus**”
- Deadline
  - **Two days before the additional demo time** at the end of this semester

Project 2: Demo

- 11/12 Thu. 9:00 ~ 18:30
- We will announce demo slots 2~3 days before.
- Tasks:
  - correct format and compile
  - QA
  - Pass test cases
  - Implement 1 extra function with limit time

Implementation

Handle Function Failures !!

- **Fork** may failed
- **Create pipe** may failed
- **Select** may failed
- **Read** may failed

Select May Failed

```
if (select(maxfd + 1, &read_set, NULL, NULL, NULL) < 0) {
        // may be interrupted by signal or other errors
        // handle error
}
for (fd = 0; fd < maxfd; ++fd) {
        if (FD_ISSET(fd, &read_set)){
                //handle fd
        }
}
```

Read May Failed

```
if (read(cli_fd, buf, BUF_SIZE) < 0) {
        // may be interrupted by signal or other errors
        // handle error
}
```

Don’t Send Additional ‘\0’ Through Socket

- char str[] = “Hello”;
  - write(fd, buf, **sizeof**(str)) // **sizeof(str) is 6**
  - write(fd, buf, **strlen**(str)) // **OK**
- std::string str = “Hello”;
  - write(fd, str.c_str(), str.length()) // **OK**

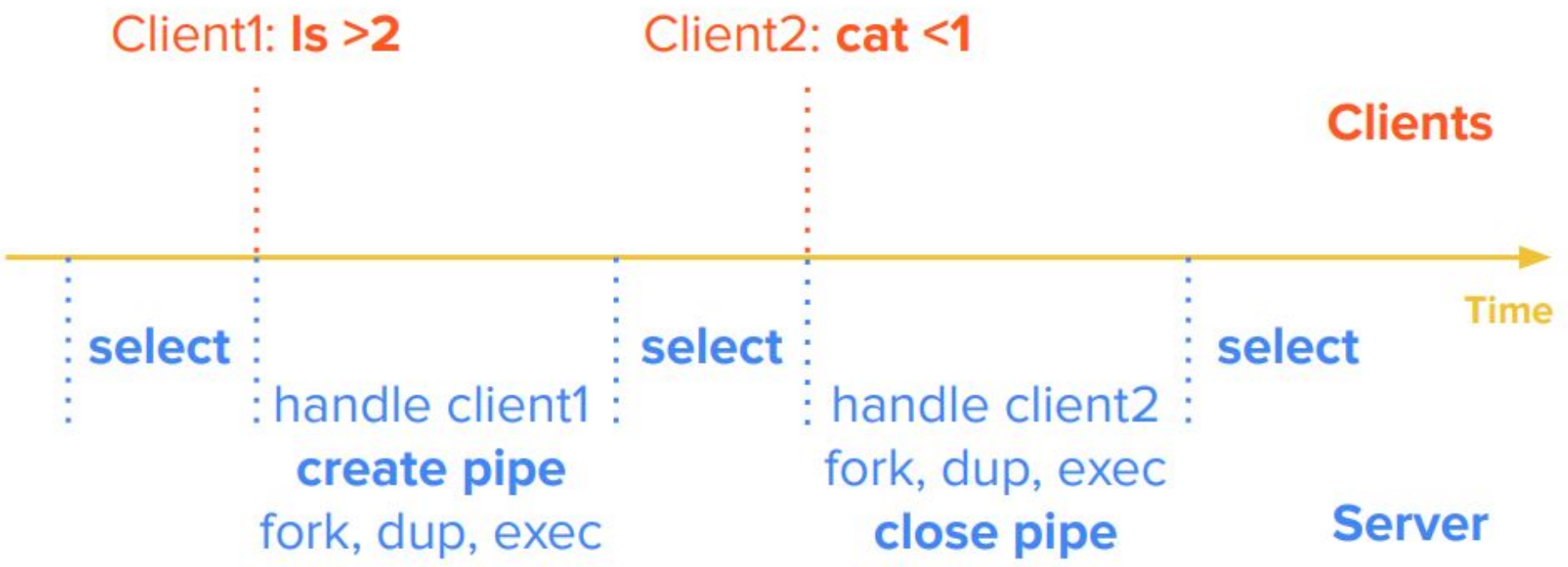
Difference between Server2 and Server3

- Server2 (np_single_proc)
  - **Single-process concurrent**
  - Use **pipe** to implement user pipe
  - Use socket to send messages directory
- Server3 (np_multi_proc)
  - **Concurrent connection-oriented**
  - Use **FIFO** to implement user pipe
  - Use **shared memory** to save **clients infos** and **messages**

Server2 (np_single_proc)

- **Single-process concurrent** (use **select**)
- Use **pipe** to implement user pipe
  - **DO NO**T use FIFO or temporary files
- Use socket to send messages directly
- Maintain environment variables for every user

Server2 (np_single_proc) - User Pipe



Server3 (np_multi_proc)

- **Concurrent connection-oriented**
- Use **FIFO** to implement user pipe
- Use shared memory to save clients infos and messages
- Handle signal
- Server3 will be terminated by SIGINT (Ctrl-C)
  - Receive SIGINT ➡ Clean up shared memory ➡ exit

Server3 (np_multi_proc) - User Pipe send

Server3 (np_multi_proc) - User Pipe recv