

Project2: RWG System



NP TA 博鈞

11/14 23:55

Project 2 Deadline

Demo: 11/17 Wed.

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

3 Servers

- 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**
- np_multi_proc (Multiple users)
 - **Project 1** + **User pipe** + **4 functions** + **Broadcast message**
 - **Concurrent connection-oriented** + **FIFO** + **Shared memory**
 - FIFO: User pipe
 - Shared memory: Broadcast message, client information

Project 2: Submission

- Create a directory named as your student ID, put all files into the directory.
- You must provide Makefile. Three executable files named **np_simple** (server 1), **np_single_proc** (server 2), **np_multi_proc** (server 3) 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. **np_simple**, noop, removetag, test.html, **.git**, **__MACOSX**)
- zip the directory and upload the .zip file to e3 platform

ATTENTION! We only accept .zip format

Project 2: Office Hour

- 11/01, 11/08 (Mon.) 10:00 - 12:00
 - Online discussion using Microsoft Teams
 - You should send an email to all the TAs to make an appointment

Project 2: Demo

- 11/17 Wed.
 - Online demo
 - Test your program and ask some questions

Project 2: About Questions

- You are **HIGHLY** encouraged to ask your questions on Microsoft Teams **Project 2 channel**.
 - Check the spec and other questions first
- For personal problems, you can mail to **all** the TAs:
 - lcd010308@gmail.com
 - kuo0404@gmail.com
 - hpc.cs08g@nctu.edu.tw
 - kyojeong.cs10@nycu.edu.tw

Scenario

Server 1

```
bash$ telnet nplinux1.cs.nctu.edu.tw 7001
% ls | cat
bin test.html
% ls |1
% cat
bin test.html
% exit
bash$
```

Server 2, 3

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

Login/Logout message

When a user login, broadcast as follows:

```
*** User '<user name>' entered from <IP>:<port>. ***
```

When a user logout, broadcast as follows:

```
*** User '<user name>' left. ***
```

Example:

```
[terminal of all users]
```

```
*** User '(no name)' entered from 140.113.215.63:1013. *** # user logins
```

```
*** User '(no name)' left. *** # user logouts
```

who - Get Information of All Users

```
% who
```

```
<ID>      <nickname>  <IP:port>   <indicate me>  
1    IamStudent   140.113.215.62:1201  <-me  
2    (no name)    140.113.215.63:1013  
3    student3     140.113.215.62:1201
```

name - Rename

```
[terminal of mine]
% name Mike
*** User from 140.113.215.62:1201 is named 'Mike'. ***
%
```

```
[terminal of all other users]

% *** User from 140.113.215.62:1201 is named 'Mike'. ***
```

If Mike is on-line, and I want to change name to Mike, this name change will fail.

```
[terminal of mine]
% name Mike
*** User 'Mike' already exists. ***
%
```

Fail

tell - Send Message to Someone

Assume my name is 'IamStudent'.

```
[terminal of mine]
% tell 3 Hello World.
%
```

If user 3 exists,

```
[terminal of user id 3]
% *** IamStudent told you ***: Hello World.
```

If user 3 doesn't exist,

```
[terminal of mine]
% tell 3 Hello World.
*** Error: user #3 does not exist yet. ***
%
```

Fail

yell - Broadcast Message

Assume my name is 'IamStudent'.

```
[terminal of mine]
```

```
% yell Good morning everyone.
```

```
*** IamStudent yelled ***: Good morning everyone.
```

```
%
```

```
[terminal of all other users]
```

```
% *** IamStudent yelled ***: Good morning everyone.
```


User Pipe

student1 (#1) pipes a command into student2(#2) via a pipe #1->#2.

```
user1 login
user2 login
% cat test.html >2
*** student1 (#1) just piped 'cat test.html >2' to student2 (#2) ***
% cat test.html >2
*** Error: the pipe #1->#2 already exists. ***
```

student2(#2) can receive from the pipe #1->#2.

```
% cat <1
*** student2 (#2) just received from student1 (#1) by 'cat <1' ***
...some output... # message from pipe #1->#2.
% cat <1
*** Error: the pipe #1->#2 does not exist yet. ***
% cat <3
*** Error: user #3 does not exist yet. ***
```

Implementation

3 Servers

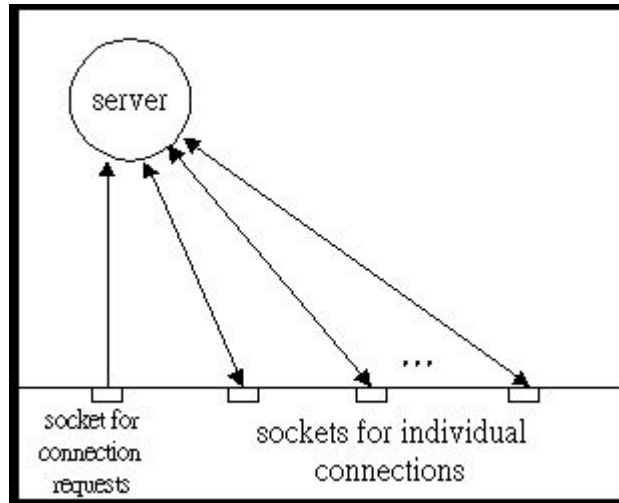
- np_simple (Single user)
 - **Project 1**
 - **Concurrent connection-oriented**
- np_single_proc (Multiple users)
 - **Project 1** + **User pipe** + **4 functions** + **Broadcast message**
 - **Single-process concurrent**
- np_multi_proc (Multiple users)
 - **Project 1** + **User pipe** + **4 functions**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

Difference between Server2 and Server3

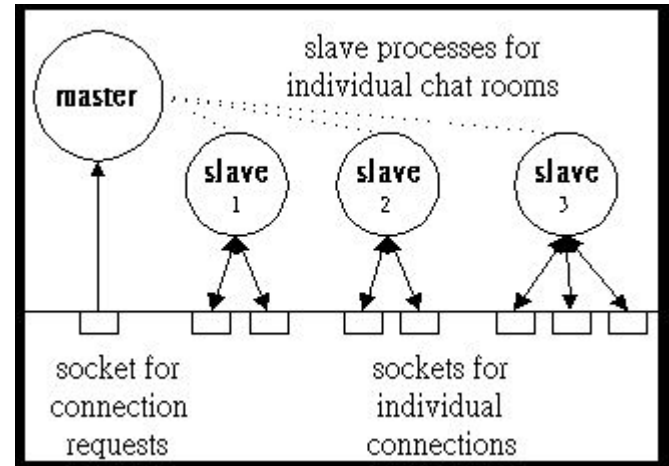
- Server2 (np_single_proc)
 - **Single-process concurrent**
 - Use **pipe** to implement user pipe
 - Use socket to send messages directory
- Server3 (np_multi_proc)
 - **Concurrent connection-oriented**
 - Use **FIFO** to implement user pipe
 - Use **shared memory** to save **clients infos** and **messages**

Client-server Model

Single-process concurrent

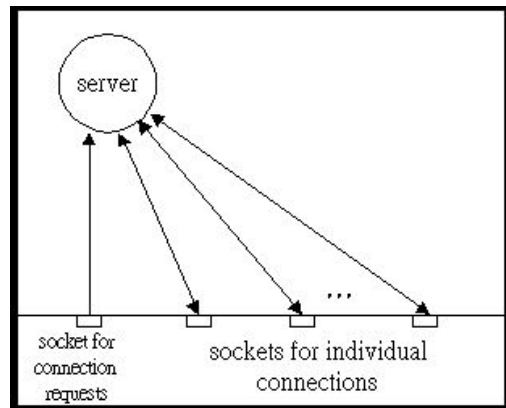


Concurrent connection-oriented

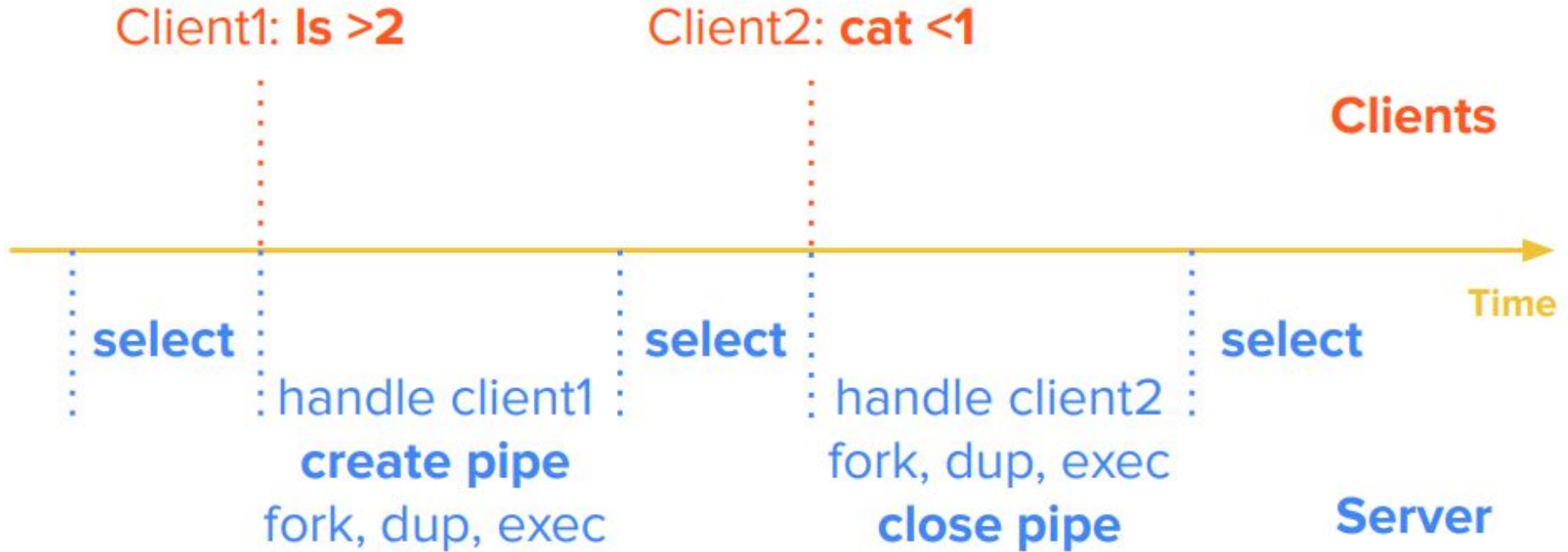


Server2 (np_single_proc)

- **Single-process concurrent** (use **select**)
- Use **pipe** to implement user pipe
 - **DO NOT** 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

Client2: **cat <1**

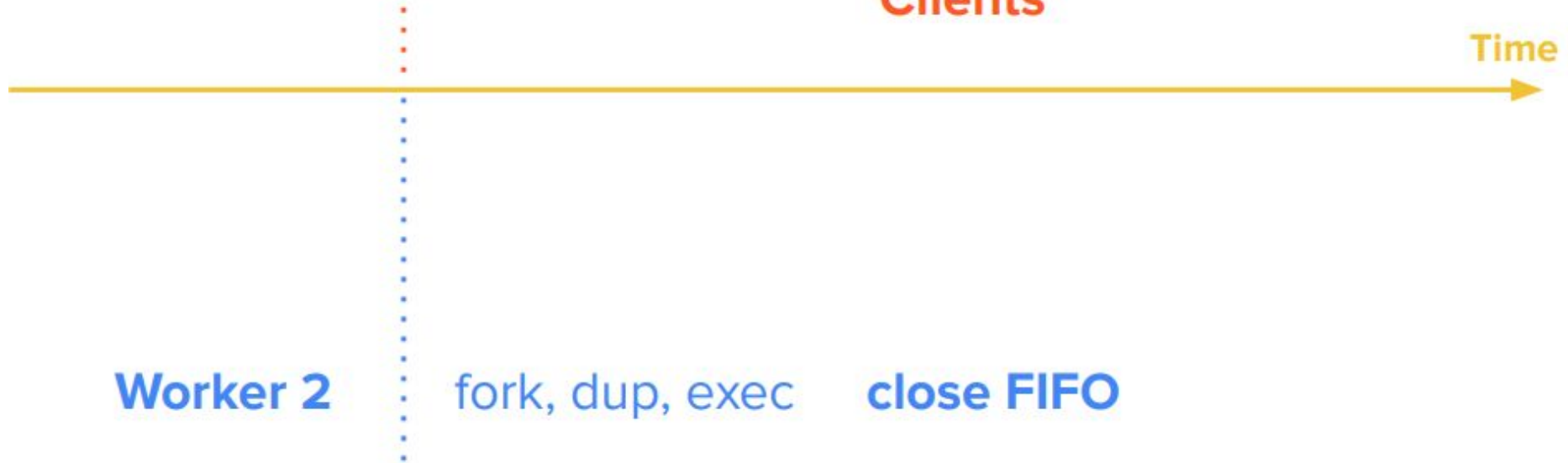
Clients

Time

Worker 2

fork, dup, exec

close FIFO



User Pipe Detail

- Pipe **stdout** only
- Whole command line should be printed in broadcast message

[terminal of user1]

```
% cat test.html | removetag0 >2
```

```
*** user1 (#1) just piped 'cat test.html | removetag0 >2' to user2 (#2)
```

```
Error: illegal tag "!test.html"           // error message from removetag0
```

```
%
```

[terminal of user2]

```
% cat <1
```

```
Test ...
```

```
%
```

User Pipe - Error Handling

- When user pipe error, each command should still be executed
 - Some command prints something itself
 - Prevent stuck when pipe large file

```
% cat test.html | removetag0 >999
```

```
*** Error: user #999 does not exist yet. ***
```

```
Error: illegal tag “!test.html”           // error message from removetag0
```

```
% Pikachu <999                          // Pikachu prints input message and Pika!Pika!
```

```
*** Error: user #999 does not exist yet. ***
```

```
Pika!Pika!
```

```
% cat LargeFile | cat | cat >999
```

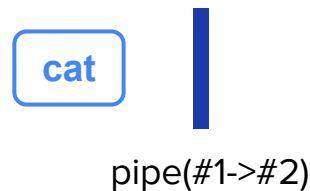
```
*** Error: user #999 does not exist yet. ***
```

User Pipe - Error Handling

% cat <2 | Pikachu



% cat LargeFile >2



% cat <999 | Pikachu // user pipe error



% cat LargeFile >999 // user pipe error



User Pipe - Error Handling

- Redirect stdin/stdout to **/dev/null**
 - stdin: EOF
 - stdout: dump everything

% cat <999 | Pikachu // user pipe error



% cat LargeFile >999 // user pipe error



Issues

Handle Function Failures !!

- **Fork** may failed (Project 1)
- **Create pipe** may failed (Project 1)
- **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  
}
```

Q&A