

컴퓨터네트워크

(과제 3 RTS, epoll)

2023. 11



학번	202102699
이름	정민경
제출일	2023.11.13

구현 1. sender / receiver 실행 결과

-코드 설명

rts_receiver 에서는 실행하면 본인의 pid 를 return 해주고, 이것을 사용해서 rts_sender 에서 인자로 receiver 의 pid 와 전달하고자 하는 value 를 전달해 rts_receiver 에서 출력하도록 구현함.

-결과 화면

```
oodscore@lambda2:~/jmink002/CN$ ./rts_receiver
My pid 323103
User RTS signal 323119
Sig Number 34
User Data is 100000
Press Enter to continue...

User RTS signal 323120
Sig Number 34
User Data is 1
Press Enter to continue...

User RTS signal 323121
Sig Number 34
User Data is 222
Press Enter to continue...

User RTS signal 323122
Sig Number 34
User Data is 1004
Press Enter to continue...

oodscore@lambda2:~/jmink002/CN$ ./rts_sender 323103 190000
oodscore@lambda2:~/jmink002/CN$ ./rts_sender 323103 1
oodscore@lambda2:~/jmink002/CN$ ./rts_sender 323103 222
oodscore@lambda2:~/jmink002/CN$ ./rts_sender 323103 1004
oodscore@lambda2:~/jmink002/CN$
```

구현 2. rts_server / rts_client 구현 (계산기)

-코드 설명

rts_server 에서는 client 에서 온 data 를 수신해 op 에 따라 결과값을 계산한 뒤 화면에 출력 후 client 로 보내는 역할을 하는 코드를 구현.

rts_client 에서는 first number, op, second number 를 std::cin 으로 입력받고, 생성한 socket 으로 data (= mycal) 을 보내도록 구현함. 그 후 server 에서 op 에 따라 계산된 결과가 다시 돌아오는 것을 recv 함수를 통해 받아서 화면에 출력해주고, "Do you want to continue" 에 대한 응답을 받아 y 이면 다시 first number, op, second number 를 입력받아 다시 server 에 보내는 식으로 작동되도록 하였고, n 이면 종료되도록 구현하였다.

-결과 화면

```
vim (vim) oodscore@lambda2: ~ (ssh) oodscore@lambda2: ~ (ssh)
oodscore@lambda2:~/jmink002/CN$ ./rts_server
signal wait
signal wait
op : +
1 + 2 = 3
signal wait
op : -
3 - 4 = -1
signal wait
op : *
5 * 6 = 30
signal wait
op : /
7 / 8 = 0
signal wait
recv error
: Success
signal wait
recv error
: Bad file descriptor
signal wait
^C
oodscore@lambda2:~/jmink002/CN$

oodscore@lambda2:~/jmink002/CN$ ./rts_client
Enter the first number: 1
Enter the operator (+, -, *, /): +
Enter the second number: 2
1 + 2 = 3
Do you want to continue (y/n)? y
Enter the first number: 3
Enter the operator (+, -, *, /): -
Enter the second number: 4
3 - 4 = -1
Do you want to continue (y/n)? y
Enter the first number: 5
Enter the operator (+, -, *, /): *
Enter the second number: 6
5 * 6 = 30
Do you want to continue (y/n)? y
Enter the first number: 7
Enter the operator (+, -, *, /): /
Enter the second number: 8
7 / 8 = 0
Do you want to continue (y/n)? n
oodscore@lambda2:~/jmink002/CN$
```

구현 3. rts_server_th / rts_client 구현 (계산기)

-코드 설명

rts_server_th.cc 에서는 serverThread 함수와, signalHandler 함수를 정의하고, main 에서 thread 생성하는 코드를 구현했다.

serverThread 함수는 Thread 가 호출되고 나서 하는 일 즉, 이번 코드에서는 data 를 client 측에서 받은 후 rts_server 처럼 op 출력하고, 계산값 출력하는 기능을 수행 하도록 구현했다. 이때 client 로부터 수신할 때는 N2H 함수를 써서 형을 변환해주었고, 결과를 다시 client 로 보낼 때 H2N 함수를 사용해 다시 형변환해주고 send 해주 었다.

signalHandler 함수에서는 어느 기능을 수행해야할지 잘 몰라 std::cout 으로 무슨 번 호의 signal 을 받았는지 출력하도록 구현하였다.

rts_client 는 구현 2와 동일함.

-결과 화면

```
oodscore@lambda2:~/jmink002/CNS$ ./rts_server_th
into serverThread
op : +
1 + 2 = 3
op : /
6 / 2 = 3
op : -
7 - 0 = 7
op : *
6 * 6 = 36
recv error: Success
^C
oodscore@lambda2:~/jmink002/CNS$

oodscore@lambda2:~/jmink002/CNS$ ./rts_client
Enter the first number: 1
Enter the operator (+, -, *, /): +
Enter the second number: 2
1 + 2 = 3
Do you want to continue (y/n)? y
Enter the first number: 6
Enter the operator (+, -, *, /): /
Enter the second number: 2
6 / 2 = 3
Do you want to continue (y/n)? y
Enter the first number: 7
Enter the operator (+, -, *, /): -
Enter the second number: 0
7 - 0 = 7
Do you want to continue (y/n)? y
Enter the first number: 6
Enter the operator (+, -, *, /): *
Enter the second number: 6
6 * 6 = 36
Do you want to continue (y/n)? n
oodscore@lambda2:~/jmink002/CNS$
```

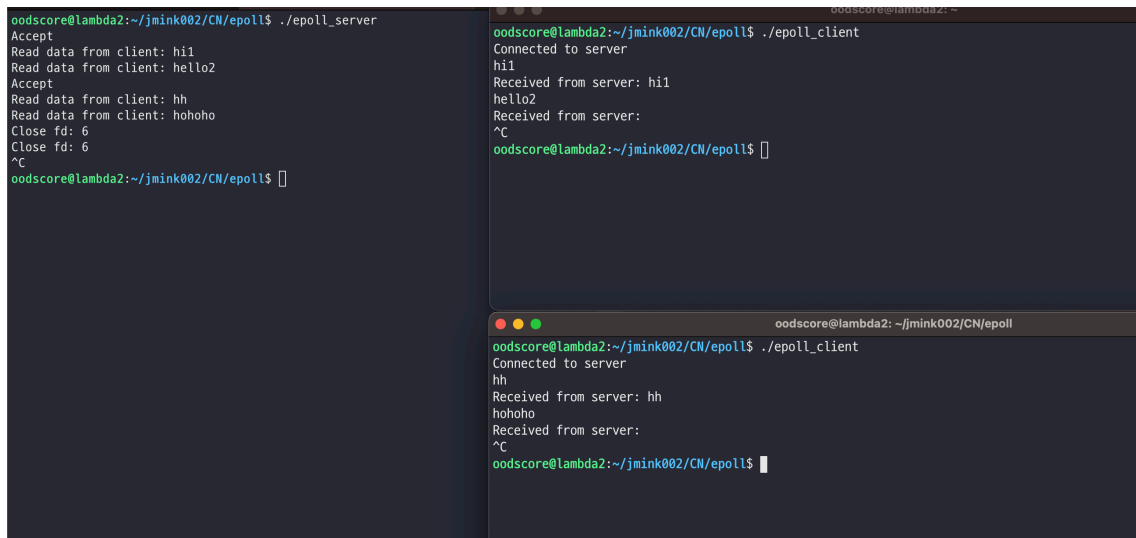
구현 4. epoll_server / client 구현

- 코드 설명

epoll_server.cc 에서는 epoll_create 함수를 사용해서 epoll 을 생성해 efd 에 저장한다. 그 후 epoll_ctl 함수를 사용해서 이벤트 폴에 생성한 efd 를 추가한 후, epoll_wait 함수를 사용해서 이벤트를 기다리는 방식으로 구현했다. 그 후 이벤트가 발생했다면 루프를 돌면서 처리한다.

epoll_client.cc 는 우리가 기존에 구현하던 client 와 동일한 방식으로 구현이 되어있고, server 와 연결되어 client 역할을 수행한다.

- 결과 화면



```
oodscore@lambda2:~/jmink002/CN/epoll$ ./epoll_server
Accept
Read data from client: hi1
Read data from client: hello2
Accept
Read data from client: hh
Read data from client: hohoho
Close fd: 6
Close fd: 6
^C
oodscore@lambda2:~/jmink002/CN/epoll$

oodscore@lambda2:~/jmink002/CN/epoll$ ./epoll_client
Connected to server
hi1
Received from server: hi1
hello2
Received from server:
^C
oodscore@lambda2:~/jmink002/CN/epoll$

oodscore@lambda2:~/jmink002/CN/epoll$ ./epoll_client
Connected to server
hh
Received from server: hh
hohoho
Received from server:
^C
oodscore@lambda2:~/jmink002/CN/epoll$
```