2012011658 김진권

1.Gprof client2 분석

gprof의 경우 fork를 한 경우, 아래와 같은 처리를 해줘야 한다.

1. GMON\_OUT\_PREFIX environment 설정

export GMON\_OUT\_PREFIX=gmon

을 설정하여, gmon.out 파일 이름이 gmon.pid로 설정되게 해준다.

해당 설정을 안해주게 되면, 부모 gmon.out과 자식 gmon.out이 덮어쓰게 되어지게 된다.

1. 자식 profiling setting

if ((pid = fork()) > 0)

{

input\_and\_send(s\_sock, servaddr, argv[3]);

}else if (pid == 0)

{

extern void \_start (void), etext (void);

monstartup ((u\_long) &\_start, (u\_long) &etext);

recv\_and\_print(s\_sock);

}

와 같이 자식 process의 profiling을 setting 해준다.

1. UDP\_server code 오류 수정.

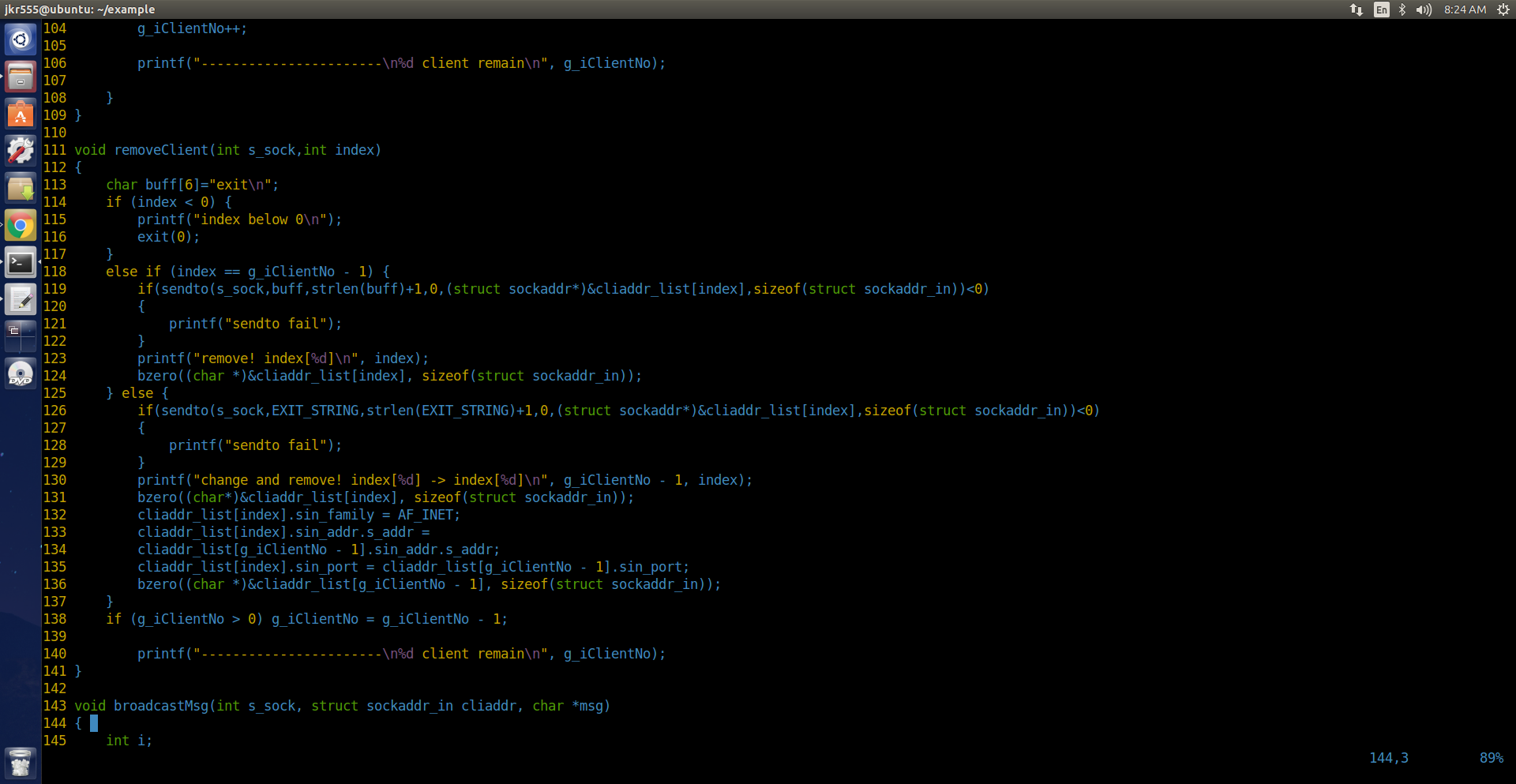
if (strstr(buf, EXIT\_STRING) != NULL )

break;

UDP\_client2의 자식 process의 경우 recv\_and\_print 함수를 담당하게 되는데, 해당 process가 종료될려면 EXIT\_STRING을 recvfrom으로 읽어 들어와야 한다.

따라서, server에서 EXIT\_STRING이 입력받아진 경우 해당 client로 EXIT\_STRING을 전송해야만 UDP\_client2의 부모(input\_and\_send 담당), 자식(recv\_and\_print담당)이 올바르게 종료되며, 부모,자식 두개의 gmon.pid 파일을 얻게 된다.

UDP\_server의 removeClient를 다음과 같이 수정하면 올바르게 동작한다.



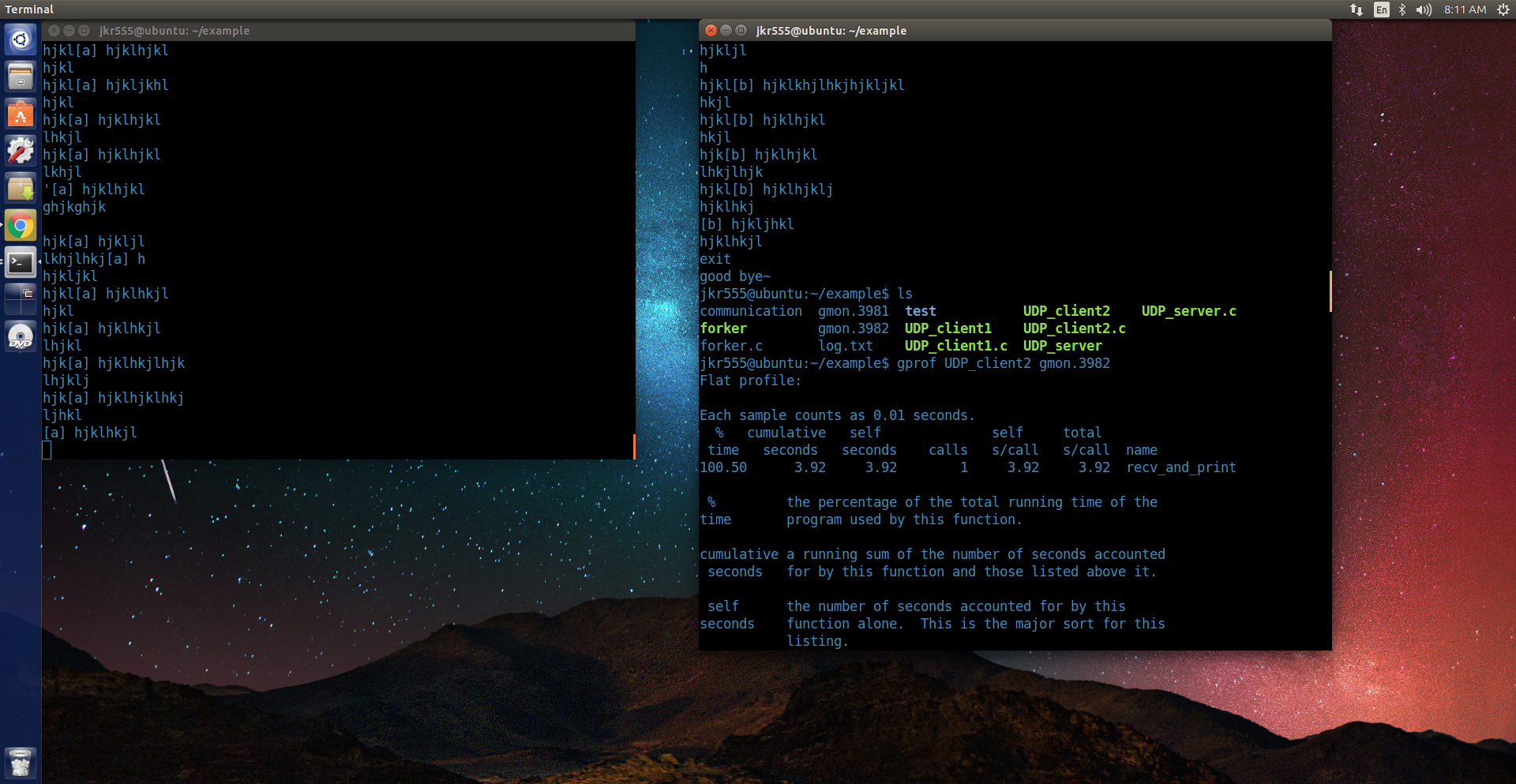
Flat profile

함수 수행이 너무 빨라. 시간이 제대로 측정되지 않아서

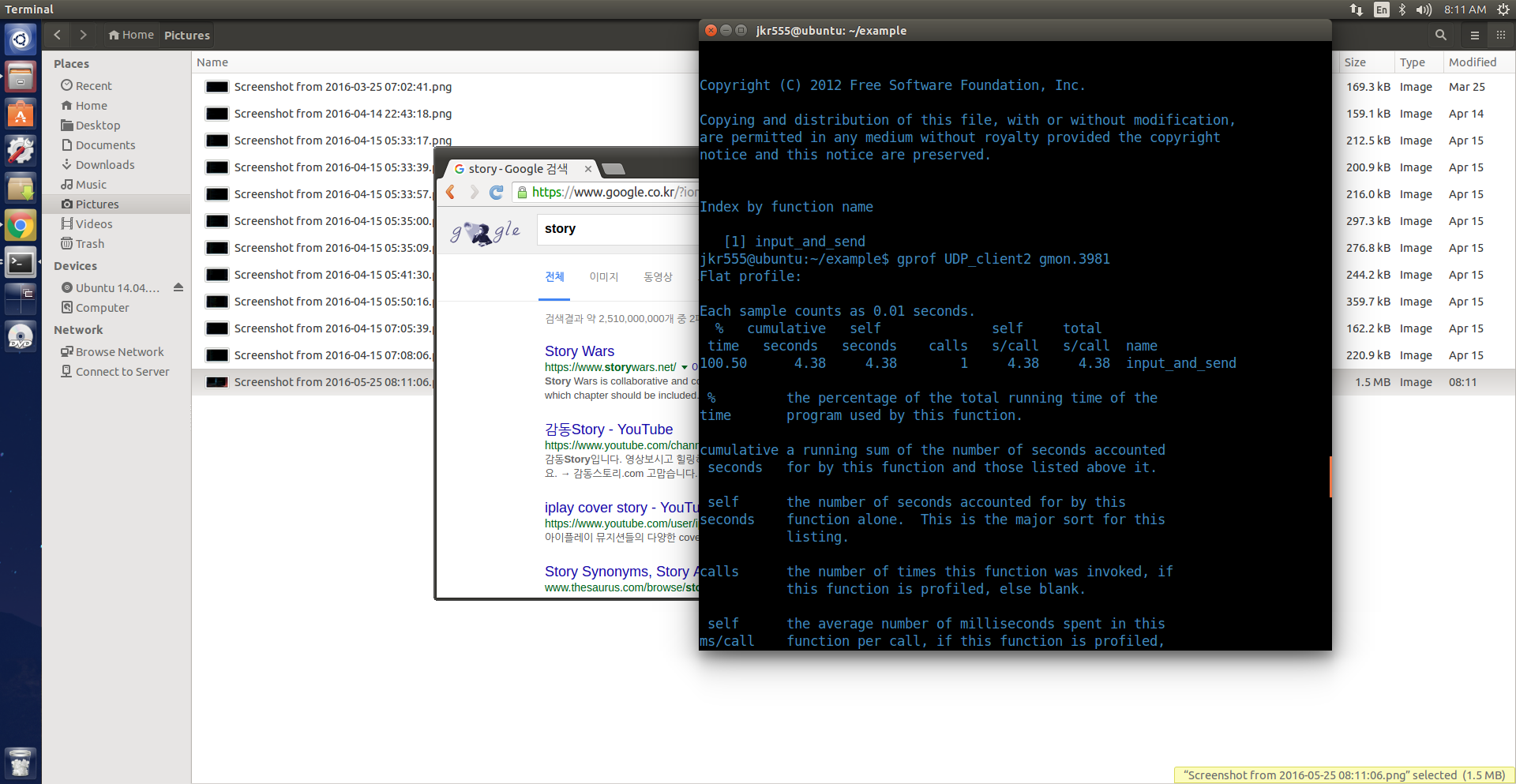
recv\_and\_print & input\_and\_send 함수 각각에서

for(i=0;i<0xfffffff;i++) 을 임의로 추가하여, 시간 측정을 하였다.

1. 자식 프로세스

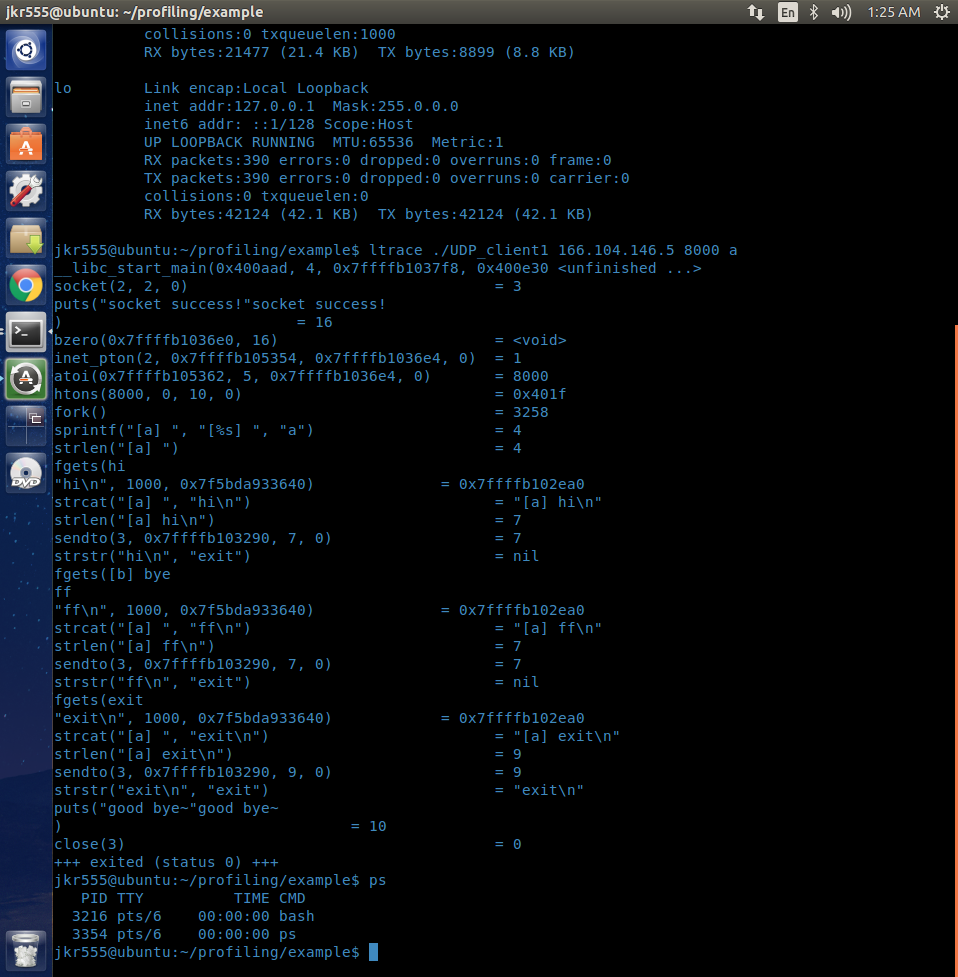


1. 부모 프로세스

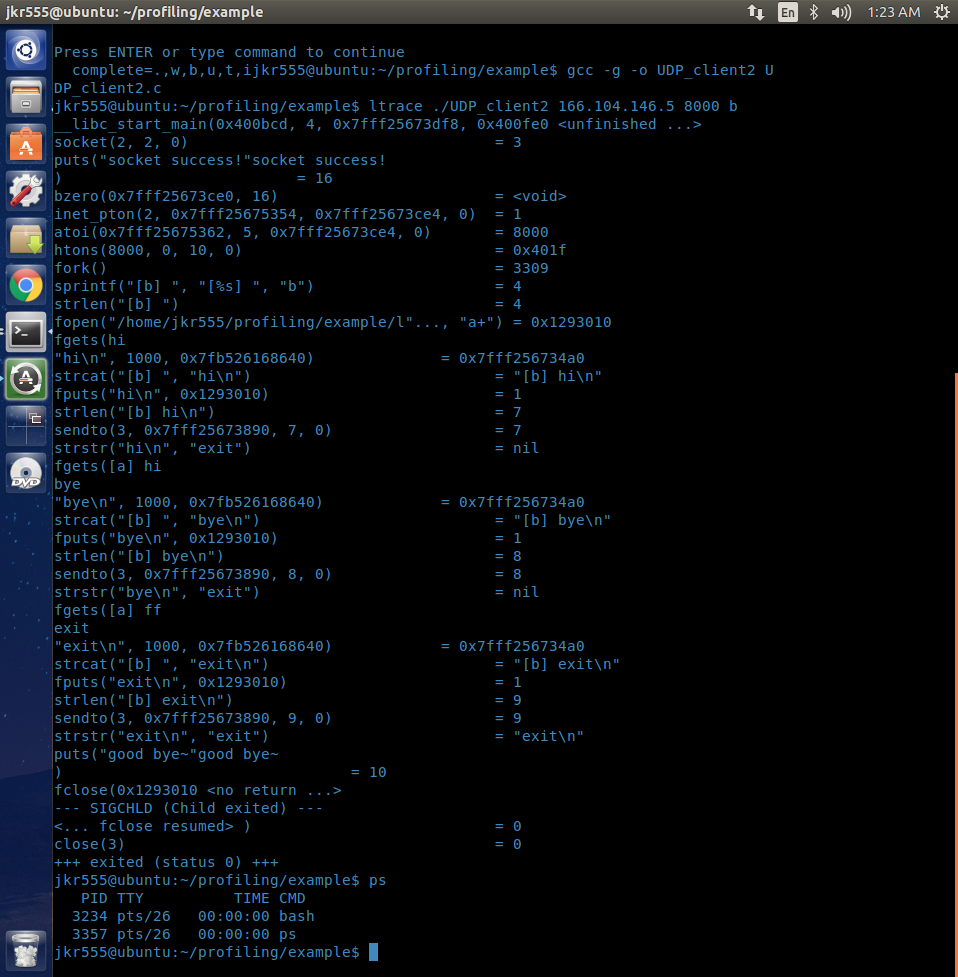


2.ltrace client1, client2 분석

1. client1



2.client2



Client1와 client2는 둘다, 소켓을 열은 후, 자신의 문자열을 fegts로 얻은 후, sendto로 전송한다. 하지만 client2 의 경우 해당 문자열을 fputs을 이용하여 사전에 열은 파일에 write을 추가적으로 수행한다.