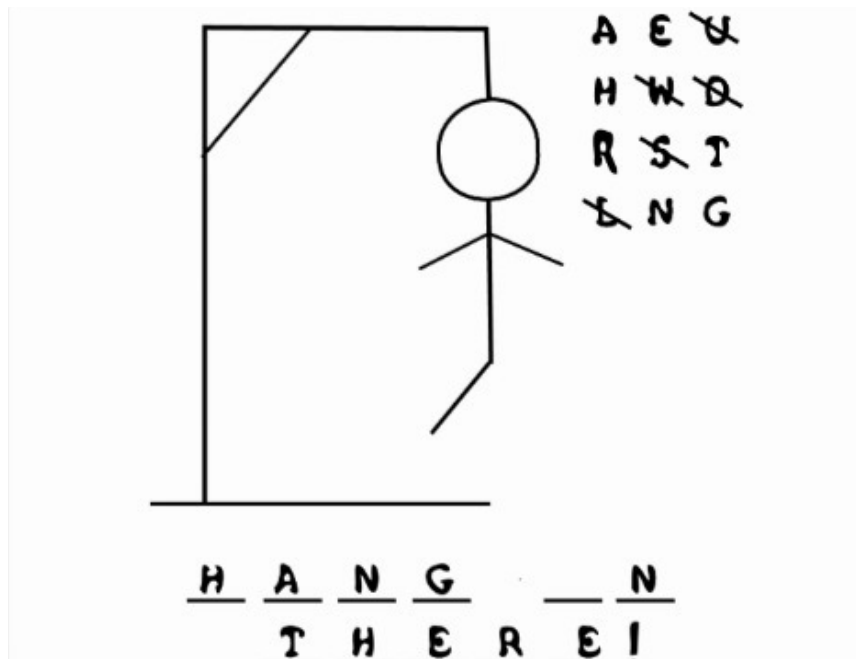


# OpenSource Programming

## Team Project : HangMan Game

### 프로젝트 주제



저희 팀의 주제는 “**Socket을 이용한 Server와 Client 간의 Hang Man Game 구현**” 입니다. 먼저 Hang Man Game 의 규칙은 이렇습니다. 출제자가 처음 영어 단어를 제시하면 답변자에게는 글자 수 만큼의 \* 이 보이게 됩니다. 답변자는 알파벳을 하나씩 입력하며 정답을 알아맞히는데, 그 단어를 구성하는 알파벳을 입력한 경우에는 그것이 있는 자리가 모두 보이게 됩니다. 틀릴 경우에는 사람이 머리-팔-몸통-다리 순으로 하나씩 그려지고 사람이 완성될 경우 교수형을 당하는 그림이 완성되어 게임에서 탈락하게 됩니다. 이 게임을 구현하기 위한 함수와 시스템 호출에 대해 공부해보고 Hang Man Game을 완성하는 것이 저희 팀의 목표이자 주제입니다.

## 구현 과정

이 게임을 구현하기 위해서 고려해야할 주요한 사항들이 몇 가지 있었습니다. 첫 번째, 답변자와 출제자가 있기 때문에 프로세스를 두 개 생성해야 한다는 것. 두 번째, 프로세스 간의 통신이 가능해야 한다는 것. 세 번째, read와 write를 동시에 수행해야 한다는 것입니다. 프로세스를 두 개 만들어 그것들 간의 통신을 가능하게 하기 위해서 socket을 사용하기로 하였습니다. 이로써 출제자 hangmanserver.c와 답변자인 hangmanclient.c 코드를 짜게 되었습니다. 또한 client 안에서 read와 write를 동시에 해야하는 문제가 있었는데, thread나 pipe를 사용하는 경우도 생각해보았으나, 색다른 방법을 시도해보고 싶어 fork를 통해 이를 구현해보기로 하였습니다.

기본적인 주제 선정과 틀을 잡고 난 뒤, socket에 대한 공부를 하였습니다. 수업시간에 배운 것이기는 하지만 아직 많이 생소했기에 개인적으로 공부하는 시간을 통해 개념을 확실히 잡기로 하였습니다. 각자 어느 정도 socket에 대해 알아본 다음 기본 틀을 세웠습니다.

### 기본 소켓통신 파일

dbs1597 committed 2779d3a 2 changed files

hangmanclient.c		@@ -1,13 +1,51 @@
hangmanserver.c	1	1 #include <stdio.h>
	2	2 #include <sys/socket.h>
	3	3
	4	4 -int main(argc, *argv[]) {
		4 +#include <stdio.h>
		5 +#include <stdlib.h>
		6 +#include <string.h>
		7 +#include <unistd.h>
		8 +#include <arpa/inet.h>
		9 +#include <sys/types.h>
		10 +#include <sys/socket.h>
		11 +
		12 +#define IP_ADDR "192.168.43.175"
		13 +#define PORT 9999
		14 +
		15 +void error_handling(char *message);
		16 +
		17 +int main(int argc, char* argv[])
		18 +{
		19 + int sock;
		20 + struct sockaddr_in serv_addr;
		21 + char message[30];
	5	22
	6	23 - //socket1
		23 + sock=socket(PF_INET, SOCK_STREAM, 0);
		24 + if(sock == -1)
		25 + error_handling("socket() error");
	7	26

클라이언트와 서버가 통신하는 단계에 따라 기본적인 시스템 호출을 정리했습니다.

중간 과정으로, fork로 read와 write를 구현 해보기 전 thread를 이용해 입출력 동시에 하기를 시도해보았습니다.

## wrt function

JeongUk Kim committed 4c4952f 1 changed file

wrt function 생성

write 부분 수정함

hangmanclient.c		
		@@ -5,6 +5,7 @@
5	5	#include <arpa/inet.h>
6	6	#include <sys/types.h>
7	7	#include <sys/socket.h>
	8	+#define BUF_SIZE 50
8	9	
9	10	void error_handling(char *message);
10	11	
		@@ -35,13 +36,7 @@ int main(int argc, char* argv[])
35	36	puts("Connecting success!!!");
36	37	
37	38	//write
38	-	message[0] = 0;
39	-	while(strcmp(message, "exit") != 0)
40	-	{
41	-	printf("Enter a message : ");
42	-	fgets(message, sizeof(message), stdin);
43	-	write(sock, message, strlen(message));
44	-	}
	39 +	wrt((void *) &sock);
45	40	//close
46	41	close(sock);
47	42	return 0;
		@@ -53,3 +48,14 @@ void error_handling(char *message)
53	48	fputc('\n', stderr);
54	49	exit(1);
55	50	}
	51 +	
	52 +	void wrt(void * arg) {
	53 +	
	54 +	int sock = *((int *)arg);
	55 +	int sock = *((int *)arg);
	56 +	char input[BUF_SIZE];

thread 이용 시 실행할 write 기능의 함수를 짰 모습입니다.

## player1,2

minju committed 8407a64 1 changed file

hangmanserver.c		
		@@ -20,6 +20,9 @@ int main(int argc, char *argv[])
20	20	
21	21	char message[MAXLEN]="Hello World!";
22	22	
	23 +	int usercount = 0;
	24 +	int user[2] = {0};
	25 +	
23	26	//socket
24	27	serv_sock=socket(PF_INET, SOCK_STREAM, IPPROTO_TCP); //양방향통신
25	28	if(serv_sock == -1)
		@@ -54,6 +57,22 @@ int main(int argc, char *argv[])
54	57	error_handling("accept() error");
55	58	puts("Server] Accepted!!!");
56	59	
	60 +	usercount++;
	61 +	
	62 +	if(usercount<=2){ // player 2명 받기
	63 +	user[usercount-1]=clnt_sock;
	64 +	
	65 +	send(clnt_sock,"Welcome to HangmanGame!\n",strlen("Welcome to HangmanGame!\n"),0);
	66 +	
	67 +	if(usercount==1) //player 1
	68 +	send(clnt_sock,"상대방이 접속하면 게임이 시작됩니다.\n");
	69 +	else if(usercount==2){ //player 2
	70 +	send(clnt_sock,"잠시 후 게임이 시작됩니다.\n",strlen("잠시 후 게임이 시작됩니다.\n"),0);
	71 +	}
	72 +	
	73 +	}
	74 +	else // player 2명 다 차면
	75 +	send(clnt_sock,"정원이 다 찹니다.\n",strlen("정원이 다 찹니다.\n"),0);
57	76	message[0] = 0;
58	77	idx = 0;
59	78	while(read_len=read(clnt_sock, &message[idx], 1))

에러

whdbeka committed 941636f 1 changed file

hangmanserver.c			@@ -74,7 +74,7 @@ int main(int argc, char *argv[])
	74	74	message[0] = 0;
	75	75	idx = 0;
	76	76	
	77	-	while(read_len=read(clnt_sock, &message[idx], 1))
		77 +	while(read_len==read(clnt_sock, &message[idx], 1))
	78	78	{
	79	79	if(read_len==-1){
	80	80	error_handling("read() error!");

중간과정의 커밋들 입니다.

1인 행맨-5(최종)

dbs1597 committed 484f8f4 1 changed file

hangmanserver.c			@@ -16,14 +16,14 @@ int main(int argc, char *argv[])
	16	16	int serv_sock;
	17	17	int clnt_sock;
	18	18	int read_len , idx;
	19	-	int i, j, incorrect, chance, last;
		19 +	int i, j, incorrect, chance, last, count;
	20	20	struct sockaddr_in serv_addr;
	21	21	struct sockaddr_in clnt_addr;
	22	22	socklen_t clnt_addr_size;
	23	23	
	24	24	char word[MAXLEN], question[MAXLEN];
	25	25	
	26	-	char message[MAXLEN];
		26 +	char message[10];
	27	27	
	28	28	if(argc!=2) {
	29	29	printf("Usage : %s <port>\n", argv[0]);
			@@ -57,6 +57,7 @@ int main(int argc, char *argv[])
	57	57	incorrect = 0;
	58	58	chance = 0;
	59	59	last = 0;
		60 +	count = 0;

마무리단계 커밋 제출을 한 모습입니다.

## 코드 분석

**hangmanclient.c** 에서 socket 생성과 connect 후 fork를 통해 생성된 자식 프로세스에서는 server에서부터 수신된 메시지를 read해 화면에 출력시키고 quit일 경우 종료시키는 역할을 합니다. 부모 프로세스에서는 키보드 입력을 fgets 함수로 받아 server로 write해 송신하는 역할을 합니다. 이로써 server로 부터 read 하는 것과 그것에 write 하는 수행을 동시에 할 수 있게 되었습니다.

### <hangmanclient.c 코드>

```
//socket
sock=socket(PF_INET, SOCK_STREAM, 0);
if(sock == -1)
    error_handling("socket() error");

memset(&serv_addr, 0, sizeof(serv_addr));
serv_addr.sin_family=AF_INET;
serv_addr.sin_addr.s_addr=inet_addr(argv[1]);
serv_addr.sin_port=htons(atoi(argv[2]));

//connect
if(connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
    error_handling("connect() error!");
puts("Connecting success!!!");

//write
fork_ret = fork();
socket 생성 후, connect한 다음 fork를 합니다.
```

```
//write
fork_ret = fork();
if(fork_ret > 0)
{
    // 부모 프로세스는 키보드 입력을 서버로 송신
    while(fgets(sendline, BUF_SIZE, stdin) != NULL)
    {
        size = strlen(sendline);
        if(write(sock, sendline, strlen(sendline)) != size)
        {
            printf("Error in write. \n");
        }
        if(strstr(sendline, "quit") != NULL) // 종료 문자열
        {
            printf("Good byte.\n");
            close(sock);
            while(1); //자식프로세서가 죽을때까지 블로킹
        }
    }
}
```

```
else if(fork_ret == 0)
{
    // 자식 프로세스는 서버로부터 수신된 메시지를 화면에 출력
    while(1)
    {
        if((size = read(sock, recvline, BUF_SIZE)) < 0)
        {
            printf("Error if read. \n");
            close(sock);
            return 0;
        }
        recvline[size] = '\0';
        if(strstr(recvline, "quit")!=NULL) // 종료 문자열
        {
            write(sock, "quit", strlen("quit"));
            break;
        }
        printf("%s", recvline); // 화면 출력
    }
}
```

각각 부모 프로세스와 자식 프로세스에서 read와 write를 하는 모습입니다.

**hangmanserver.c**에서는 socket 생성, bind, listen 후에 정답 단어를 입력하게 하고, accept 되면 client로부터 알파벳 한 글자를 읽어옵니다. 입력한 알파벳이 정답안 단어를 구성하는 경우 \* 처리된 부분을 그 알파벳으로 replace 하여 보냅니다.

<hangmanclient.c 코드>

```
serv_sock=socket(PF_INET, SOCK_STREAM, 0);
if(serv_sock == -1)
    error_handling("socket() error");

memset(&serv_addr, 0, sizeof(serv_addr));
serv_addr.sin_family=AF_INET;
serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
serv_addr.sin_port=htons(atoi(argv[1]));

if(bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
    error_handling("bind() error");

if(listen(serv_sock, 5)==-1)
    error_handling("listen() error");

clnt_addr_size=sizeof(clnt_addr);

printf("정답단어를 입력하세요.\n");
scanf("%s", word);

for(i=0; i<strlen(word); i++)
    question[i]='*';
question[i]='\0';

incorrect = 0;
chance = 0;
last = 0;
count = 0;

while(1)
{
    puts("Server] Listening...");
    clnt_sock=accept(serv_sock, (struct sockaddr*)&clnt_addr,&clnt_addr_size);

    if(clnt_sock==-1)
        error_handling("accept() error");
    puts("Server] Accepted!!");
```

socket 생성, bind, listen 후 정답단어 입력하고 accept하는 부분입니다.

```
while(read(clnt_sock, &message[0], 1))
{
    count++;
    if (count%2 == 0)
        continue;

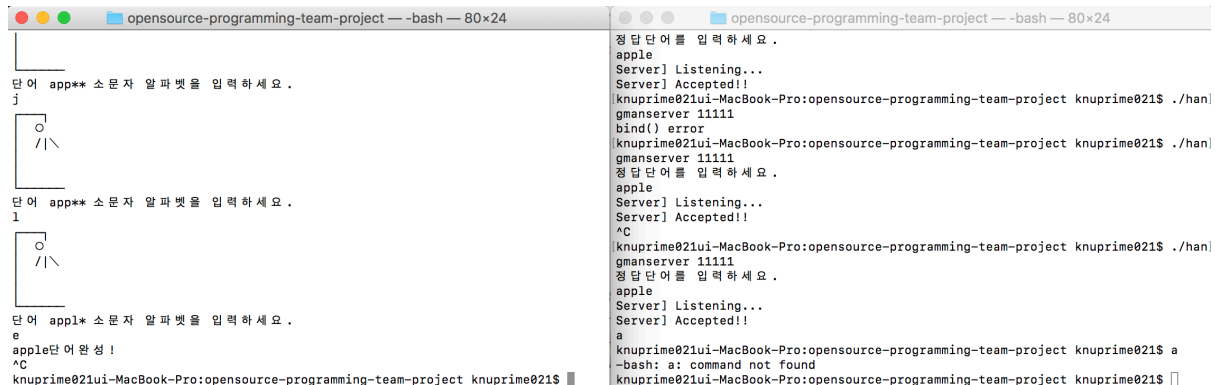
    if(read_len==-1){
        error_handling("read() error!");
        exit(1);
    }

    if(strlen(message)==1)
    {
        for(i=0; i<strlen(word); i++)
        {
            if(word[i]==message[0] && question[i] != message[0])
            {
                incorrect++;
                question[i]=word[i];

                if (incorrect == strlen(word))
                {
                    send(clnt_sock, word, strlen(word), 0);
                    send(clnt_sock, " 단어완성!\n", strlen(" 단어완성!\n"), 0);
                    close(clnt_sock);
                    close(serv_sock);
                    return 0;
                }
            }
        }
    }
}
```



## 실행화면



```
opensource-programming-team-project -- -bash -- 80x24
단어 app** 소문자 알파벳을 입력하세요 .
j
o
/ \
단어 app** 소문자 알파벳을 입력하세요 .
1
o
/ \
단어 appl* 소문자 알파벳을 입력하세요 .
e
apple 단어 완성 !
^C
knuprime021ui-MacBook-Pro:opensource-programming-team-project knuprime021$

opensource-programming-team-project -- -bash -- 80x24
정답 단어를 입력하세요 .
apple
Server] Listening...
Server] Accepted!!
knuprime021ui-MacBook-Pro:opensource-programming-team-project knuprime021$ ./han
gmanserver 11111
bind() error
knuprime021ui-MacBook-Pro:opensource-programming-team-project knuprime021$ ./han
gmanserver 11111
정답 단어를 입력하세요 .
apple
Server] Listening...
Server] Accepted!!
^C
knuprime021ui-MacBook-Pro:opensource-programming-team-project knuprime021$ ./han
gmanserver 11111
정답 단어를 입력하세요 .
apple
Server] Listening...
Server] Accepted!!
a
knuprime021ui-MacBook-Pro:opensource-programming-team-project knuprime021$ a
-bash: a: command not found
knuprime021ui-MacBook-Pro:opensource-programming-team-project knuprime021$
```

## 아쉬운 점

socket 을 이용한 통신에 익숙하지 않았기 때문에 구현을 시작하기 전에 공부를 많이 해야 했습니다. 개념 공부에 많은 시간을 할애해 구현을 할 수 있었던 시간이 줄어든 것이 아쉽습니다. 또한 깃허브에 익숙하지 않아서 협업하는 과정을 익히는 것도 서툴렀습니다. 이 프로젝트에서 배운 것을 토대로 하여 다음엔 기능을 추가시켜 더욱 완성도 있는 결과물을 내고싶다는 욕심이 생겼습니다.

## 소감

다들 구현하는 것이 익숙하지 않은 주제였지만 협업을 통해 결과물을 냈다는 것이 뿌듯합니다. 다음은 조원들의 소감입니다.

윤상현 : 소켓 통신이라는 자체가 꽤 난이도 있는 주제이다. 이 주제를 가지고 프로그램을 짤 때 처음엔 어려움이 많았다. 하지만 계속 여기저기 찾아보고 참고하면서 코드를 짜고 다듬고 하다 보니 결국 괜찮은 결과물이 나온것 같다. 물론 원래 계획에는 약간 못 미치는 결과물을 남긴 했지만, 이 과정 자체가 유익한 경험이라고 생각한다.

조유담 : 수업에서 배운 리눅스 체계와 시스템 호출 등을 자세히 익힐 수 있었고, 깃허브와 같은 오픈소스 공유 플랫폼을 직접 개발과 협업에 활용할 수 있는 좋은 기회였다. 서로 찾은 정보를 바탕으로 게임을 구현하여 재미와 흥미가 있었다.

김정욱 : github를 본격적으로 사용해본 것은 처음이었는데 협업하기에 굉장히 편리했습니다. 배운 것을 토대로 새로운 것을 완성시키는 과정이 흥미로웠습니다. socket에 익숙하지 않았지만 하다보니 재미있었고 더 잘하고 싶다는 욕심도 생겼습니다. process간의 통신에 대해 더 잘 알 수 있었습니다. 다음에도 이런 기회가 있다면 참여할 것입니다.

시민주 : 이번 프로젝트를 하면서 GITHUB를 통해 프로그래밍 협력을 하는 것을 배울 수 있었습니다. 처음에는 어렵고 생소하기도 했지만 프로젝트를 완성하는 과정에서 협력의 중요성을 느낄 수 있었습니다. 또한 수업시간에 배운 소켓을 통해서 서버와 프로세스간의 통신을 게임으로 풀어내는 과정에서 다시 한번 내용을 복습하고, 구현할 수 있는 기회가 되어 유익한 시간이었습니다.



