운영체제

리눅스 프로젝트

학과: 빅데이터
학번: 20195275
이름: 한 대 현
담당교수: 김의직
담당조교: 이상우
실험날짜: 2021년 6월 2일
제출날짜: 2021년 6월 3일

한림대학교

목차

01
실험목적

리눅스 기반으로 원카드 게임을 설계한다.

기존의 2인 게임을 3인으로 만든다.

fork() pipe를 이용한다.

무승부 규칙을 변경한다.

02 추진절차

먼저 전체적인 틀을 만들기 위해 플레이어 3를 만든다.
만들고 나면 세부적 사항인 규칙을 변경해준다.
변경하고 나서 각 플레이어 소스코드에 fork(), pipe를 달아준다.
먼저 플레이어 3 의 메세지 큐 key id, pid코드이다.

```
// 메세지 큐 key IDs
int msqid_p1_down; // manager -> player1
int msqid_p1_up; // player 1 -> manager
int msqid_p2_down;// manager -> player2
int msqid_p2_up; // player 1 -> manager2

int msqid_p3_down;// manager -> player3
int msqid_p3_up; // player 1 -> manager3


// 플레이어들의 pid;
int p1_pid;
int p2_pid;

int p3_pid;
```

02 추진절차

플레이어3의 id 를 추가해주고 메세지 큐를 만들어주는 부분이다

```
make_cards();
// 카드 셔플
shuffle(cards, 52);
printf("₩n@@ cards setting---------------------------------₩n");
// message queue ID 확인
key_t key_p1_down = 50001;
key_t key_p1_up = 50002;
key_t key_p2_down = 60001;
key_t key_p2_up = 60002;
key_t key_p3_down = 70001;
key_t key_p3_up = 70002;

// messge queue create
if((msqid_p1_down=msgget(key_p1_down,IPC_CREAT|0666))==-1){return -1;}
if((msqid_p1_up=msgget(key_p1_up,IPC_CREAT|0666))==-1){return -1;}
if((msqid_p2_down=msgget(key_p2_down,IPC_CREAT|0666))==-1){return -1;}
if((msqid_p2_up=msgget(key_p2_up,IPC_CREAT|0666))==-1){return -1;}

if((msqid_p3_down=msgget(key_p3_down,IPC_CREAT|0666))==-1){return -1;}
if((msqid_p3_up=msgget(key_p3_up,IPC_CREAT|0666))==-1){return -1;}
// messge queue reset
msgctl(key_p1_down, IPC_RMID, NULL);
msgctl(key_p1_down, IPC_RMID, NULL);
msgctl(key_p1_down, IPC_RMID, NULL);
msgctl(key_p1_down, IPC_RMID, NULL);

msgctl(key_p1_down, IPC_RMID, NULL);
msgctl(key_p1_down, IPC_RMID, NULL);
// messge queue create
if((msqid_p1_down=msgget(key_p1_down,IPC_CREAT|0666))==-1){return -1;}
if((msqid_p1_up=msgget(key_p1_up,IPC_CREAT|0666))==-1){return -1;}
if((msqid_p2_down=msgget(key_p2_down,IPC_CREAT|0666))==-1){return -1;}
if((msqid_p2_up=msgget(key_p2_up,IPC_CREAT|0666))==-1){return -1;}

if((msqid_p3_down=msgget(key_p3_down,IPC_CREAT|0666))==-1){return -1;}
if((msqid_p3_up=msgget(key_p3_up,IPC_CREAT|0666))==-1){return -1;}
```

02 추진절차

다음은 플레이어3에게 초기 카드정보를 전달하는 코드이다.
총 6개의 카드가 들어온다

```
struct gameInfo send_p3; // 플레이어 3에게 전달할 정보
struct gameInfo receive_p3; // 플레이어 3에게 전달받을 정보

// top에 위치한 카드 인덱스 표시
top = 0;
// 각 플레이어에게 전달할 초기 게임 정보 생성
//player 1
send_p1.num_cards = 0;
send_p1.manager_pid = getpid();
send_p1.player_pid = -1;
for (int i=top; i<6; i++){
    send_p1.cards[i]= cards[i];
    send_p1.num_cards ++;
    top ++;
}
// player 2
send_p2.num_cards = 0;
send_p2.manager_pid = getpid();
send_p2.player_pid = -1;
int i = 0;
for (i=0; i < 6; i++){
    send_p2.cards[i]= cards[i+top];
    send_p2.num_cards ++;
}
top += i+1;

// player 3
send_p3.num_cards = 0;
send_p3.manager_pid = getpid();
send_p3.player_pid = -1;
for (int i=0; i < 6; i++){
    send_p3.cards[i]= cards[i+top+1];
    send_p3.num_cards ++;
}
top += i+2;
```

02 추진절차

플레이어의 게임 정보를 pid에 전달하는 코드이다

```
top ++;
// 게임 정보 전달
printf("sending first game info₩n");
if(msgsnd(msqid_p1_down, &send_p1, sizeof(struct gameInfo), 0)==-1){return 0;};
if(msgsnd(msqid_p2_down, &send_p2, sizeof(struct gameInfo), 0)==-1){return 0;};

if(msgsnd(msqid_p3_down, &send_p3, sizeof(struct gameInfo), 0)==-1){return 0;};

// 플레이어의 정보 수신 --> 플레이어의 pid 저장
printf("receive the players pids₩n");
if(msgrcv(msqid_p1_up, &receive_p1, sizeof(struct gameInfo), 0, 0)==-1){return 1;};
if(msgrcv(msqid_p2_up, &receive_p2, sizeof(struct gameInfo), 0, 0)==-1){return 1;};

if(msgrcv(msqid_p3_up, &receive_p3, sizeof(struct gameInfo), 0, 0)==-1){return 1;};

p1_pid = receive_p1.player_pid;
p2_pid = receive_p2.player_pid;

p3_pid = receive_p3.player_pid;

// 게임 시작
```

02 추진절차

플레이어3 차례가 오면 실행될 코드이다.
즉 카드를 선택하면 오픈카드의 정보와 비교해 카드를 가지거나 아닐수도 있다.

```
////////////////////////////////////////////////////////////////
// Player 3의 차례/////////////////////////////////////////////
////////////////////////////////////////////////////////////////
kill(p3_pid, SIGUSR1);
printf("Player 3's turn₩n");
// 1. 업데이트된 게임 정보 전달 (즉, 현재 오픈카드 정보)
send_p3.open_card = open_card;
if(msgsnd(msqid_p3_down, &send_p3, sizeof(struct gameInfo), 0)==-1){return 0;};
// 2. 플레이어 3의 게임 정보 수신
if(msgrcv(msqid_p3_up, &receive_p3, sizeof(struct gameInfo), 0, 0)==-1){return 1;};
// 3. 게임 정보 업데이트 & 필요시 플레이어 1에게 새로운 카드 전달
if (receive_p3.open_card.value == open_card.value && receive_p3.open_card.suit == open_card.suit){
    //open 카드 정보가 같으면, 플레이어가 카드를 내려놓지 못한 것. --> 새로운 카드 전달.
    for(int i=0; i<receive_p3.num_cards; i++){
        send_p3.cards[i] = receive_p3.cards[i];
    }
    send_p3.cards[receive_p3.num_cards] = cards[top];
    send_p3.num_cards = receive_p3.num_cards+1;
    top ++;
    if(msgsnd(msqid_p3_down, &send_p3, sizeof(struct gameInfo), 0)==-1){return 0;};
}
else{
    //open 카드 정보가 다르면, 플레이어가 카드를 내려놓은 것. --> 현재 open 카드 정보 업데이트
    open_card = receive_p3.open_card;

}
```

02 추진절차

게임의 승패를 줄 코드이다.
여기선 카드가 13장 이상 넘어가면 해당하는 플레이어는 패배하게 되고
무승부가 났을 시 코인을 통해 0번째 코인을 얻은 사람만 승리하게 된다.

```
        }
        // 4. 게임 종료 판단
        if (receive_p3.num_cards == 0){
            kill(p3_pid, SIGINT); // 승리
            kill(p1_pid, SIGQUIT); // 패배
            kill(p2_pid, SIGQUIT); // 패배
            printf("Player 3 Win₩n");
            return(9);
        }
        if (receive_p3.num_cards > 13){
            kill(p3_pid, SIGQUIT); // 패배
            kill(p1_pid, SIGILL); // draw
            kill(p2_pid, SIGILL); // draw
            printf("Player 3 loser₩n");
            printf("Player 1,2 draw₩n");
            return(9);
        }
        if (top >= 51){
            //카드 다씀, 무승부, coin = 0 => win
            int coin = rand()&2;
            if (coin == 0){
                printf("p3 coin = 0");
                printf("₩n");
                printf("p3 = win");
                printf("₩n");
                printf("p1, p2 = lose");
                printf("₩n");
                kill(p3_pid, SIGINT); // 승리
                kill(p2_pid, SIGQUIT); // 패배
                kill(p1_pid, SIGQUIT); // 패배
            }
            else {
                printf("p3 coin = 1");
                printf("₩n");
                printf("p3 = lose");
                printf("₩n");
                printf("p1, p2 = draw");
                printf("₩n");
                kill(p3_pid, SIGQUIT); // 패배
                kill(p2_pid, SIGILL); // draw
                kill(p1_pid, SIGILL); // draw
            }
        }
```

02 추진절차

다음은 fork(), pipe 코드이다.
각각 플레이어코드에 fork, pipe를 달아줌으로써
부모프로세스의 데이터값을 chiled 프로세스가 출력만 하도록 하였다.

```
int fd[2];
pid_t pid;
char buf[MAX_BUF];

if(pipe(fd)<0){
    printf("pipe err₩n");
    exit(1);
}

if((pid = fork()) == -1){
    printf("fork error₩n");
    exit(1);
}
printf("₩n");

if(pid == 0){ //chile process
    printf("chile process₩n");
    read(fd[READ],buf,MAX_BUF);
}
else{ //parent process
// 초기 정보 수신
if(msgrcv(msgid_down, &my_info, sizeof(struct gameInfo), 0, 0)==-1){return 1;}
```

결론 도출

아래 사진처럼 코드가 잘 작동하는 것을 확인할 수 있다.