

유닉스 프로그래밍 (1분반)

(UNIX Programming)

[기말 프로젝트] 소켓을 활용한 COM vs. Player 블랙잭 게임

[목차]

1. 주제 설명-----	2
2. 시스템 설계도-----	3
3. 사용된 기술 -----	6
4. 사용 메뉴얼 -----	8
5. 사용 예 캡처 -----	10

<2인 1조>

팀원1: 학과: 융합전자공학전공, 학번: 201810888, 이름: 전세환

팀원2: 학과: 융합전자공학전공, 학번: 201810896, 이름: 한규현

1. 주제 설명

[소켓을 활용한 COM vs. Player 블랙잭 게임]

이 프로그램은 소켓을 활용하여 서버(서버)와 플레이어(클라이언트) 간의 블랙잭 게임을 수행하는 프로그램이다. 블랙잭은 카지노에서 플레이되는 인기 카드 게임 중 하나로, 플레이어와 딜러가 1:1로 카드를 뽑거나(Hit), 유지(Stand)하면서 21점을 넘지 않는 선에서 21에 가까운 값을 만든 사람이 승리하는 게임이다. 게임 룰 참고: <https://blog.naver.com/haeunforever/221893584138>

서버에 접속해 AI와 블랙잭을 플레이하기 위해 제작하여 플레이어와 컴퓨터 간의 대결로 프로그램을 설계하였다. 서버와 클라이언트 간의 데이터 교환은 인터넷 소켓을 이용하여 진행되며, 유닉스 환경에서 C언어로 구현했다.

자세한 게임 수행 방법은 다음과 같다.

1. COM(상대, 딜러)는 서버, 플레이어는 클라이언트로 구성되어 게임 수행

- COM(상대)는 구현된 블랙잭 알고리즘에 따라 자동 플레이

2. 통신 설정 완료 후 게임 시작

- 플레이어와 서버는 소켓을 통해 통신 설정을 진행
- 서버가 클라이언트의 정보(IP주소, 포트 번호)를 받았을 경우 통신 설정이 완료

3. 각자의 턴이 시작되며 번갈아가면서 Hit(뽑기), Stand(유지) 선택

- 플레이어는 자신이 가진 자본(Budget)을 입력
- 플레이어는 새로운 경기(판)가 시작할 때마다 해당 경기에 배팅 금액을 입력
- 플레이어는 카드를 뽑을(Hit) 것 인지, 유지(Stand)할 것인지 선택
- 서버는 플레이어에 선택에 맞춰 정해진 블랙잭 알고리즘을 수행

4. 21점을 넘거나(Bust), 플레이어와 서버 모두 Stand 선택 시, 턴 종료

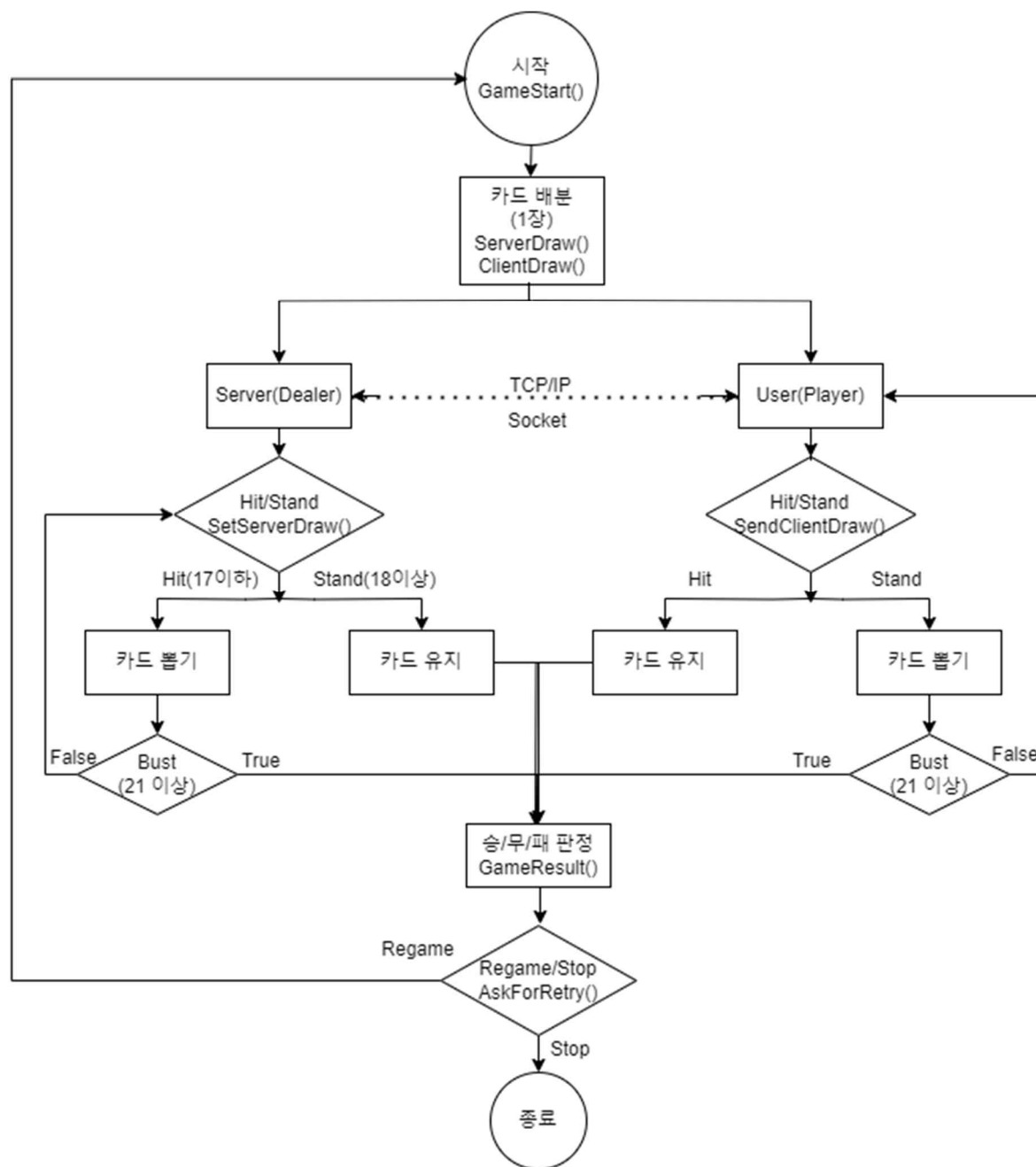
- 최종 승패 판단 및 자본 및 배팅 금액 계산은 서버에서 진행
- 서버에는 승패결과를, 플레이어에는 양쪽에 패와 승패결과를 출력

4. 경기(판) 추가 진행 혹은 종료

- 플레이어가 추가 경기를 원하면, 새롭게 배팅 후 게임 진행
- 플레이어가 추가 경기를 원하지 않거나, 파산인 경우 최종 자본 출력 후 종료

2. 시스템 설계도

1) 순서도



2) 함수 설명

함수명	함수 사용 코드	설명
SetServerSocket	Server.c	client - server간 통신 설정
ErrorCheck	Server.c, client.c	client - server간 통신 과정에서의 오류 검사
ShowInfo	Server.c	server의 콘솔 화면을 clear하고 플레이어의 자본과 배팅 금액을 출력
SetBalance	Server.c	client에서 budget을 입력 받아 balance변수에 저장하고, 초기 플레이어 자본 출력
ReceiveBet	Server.c	client에서 한 경기(판)의 배팅 금액을 입력 받아 출력
GameStart	Server.c	server에서 블랙잭 게임을 진행 server와 client가 번갈아가며 게임 진행 server나 client의 Draw state(뽑기 상태)나 Bust state(21 초과)를 확인하여 게임을 종료
SendCardClient	Server.c	server에 카드 텍 정보가 담긴 구조체를 client로 전송
SendBet	Server.c	server에서 판정한 승패에 따라 budget에서 배팅 금액을 계산하여 플레이어에게 전송 (Win: Balance+Bet, Lose: Balance-Bet, Draw: Balance)
ReceiveRetry	Server.c	플레이어가 게임을 계속할지에 대한 의견을 수신
SetServerDraw	Server.c	server가 카드를 더 뽑을지 결정
ReceiveClientDraw	Server.c	server가 플레이어의 Hit, Stand 여부를 수신
SendResult	Server.c	client로 승패 결과 전송
GameResult	Server.c	client로 server_card 구조체 전달, 승패 결정 server화면에 승패 결과 출력 (server와 client 카드의 총합으로 결정)
Shuffle	Server.c	server에서 카드 셔플, Server_Card, Client_Card, Deck 구조체 초기 상태로 초기화
ServerDraw	Server.c	server(딜러) 카드 뽑기 (Deck과 비교해 기존에 뽑은 카드면 다시 뽑음)
ClientDraw	Server.c	client(플레이어) 카드 뽑기 (Deck과 비교해 기존에 뽑은 카드면 다시 뽑음)
ServerPrintHand	Server.c, Client.c	server(딜러)가 현재까지 뽑은 카드를 출력

ClientPrintHand	Server.c, Client.c	client(플레이어)가 현재까지 뽑은 카드를 출력
ConnectToServer	Client.c	client - server간 통신 설정
GameIntro	Client.c	게임 시작 시, client 콘솔 화면 초기화 후 환영 메시지 출력
ShowBalance	Client.c	client의 자본금 출력, 배팅 금액이 적절하지 않은 경우 계속해서 배팅 금액 입력을 요청
ShowInfo	Client.c	client 게임 시작 시 화면 출력으로 초기화후 현재 자본금과 배팅금액 출력
SendBalance	Client.c	client에서 자본금을 입력 받아 server로 전송
SendClientBet	Client.c	client에서 배팅 금액을 입력 받아 server로 전송
ReceiveServerCard	Client.c	경기(판)이 끝난 후 client에 server가 뽑은 카드를 보여주기 위해 Server_Card 구조체 수신
ReceiveResult	Client.c	client에서 게임 승패 결과를 server로부터 수신
AskForRetry	Client.c	client에서 플레이어가 게임을 계속할지에 대한 의견을 서버로 전송
SendClientDraw	Client.c	server에 플레이어의 Hit, Stand 여부를 전송
ReceiveBet	Client.c	한 경기(판) 끝난 후, client가 배팅 액수가 계산된 자본금 결과 수신

3. 사용된 기술

1. 클라이언트 - 서버 간 통신

소켓 프로그래밍을 활용하여 유닉스 환경 내 클라이언트와 서버 간의 데이터 교환을 가능하게 하였다. TCP/IP 프로토콜을 기반으로 하며, 서버와 클라이언트의 IP와 PORT 정보를 통해 양방향 통신을 수행한다. 게임 진행 정보와 클라이언트의 카드 상태 등을 전송하며, 데이터의 정확한 전송을 보장하기 위해 송수신되는 패킷의 바이트 수를 체크한다. 이를 통해 패킷의 누락 없이 정확한 통신이 이루어지는지 확인하였다.

- 소켓 프로그래밍: 유닉스 시스템 내에서의 네트워크 통신을 위해 BSD 소켓을 사용하였다. 이를 통해 클라이언트와 서버 간의 연결과 데이터 전송이 가능하다.
- TCP/IP: 신뢰성 있는 데이터 전송을 위해 TCP 프로토콜을 사용하였으며, 인터넷 프로토콜(IP)을 통해 서버와 클라이언트가 통신한다.
- 데이터 통신 체크: 데이터가 송수신 될 때, 패킷의 바이트 수를 검사하여 통신 중 패킷이 올바르게 전송되었는지 확인한다.
- 오류 체크: 네트워크 연결, 소켓 생성, 데이터 수신 시 발생할 수 있는 오류를 체크한다. 오류 발생 시, 적절한 에러 메시지를 출력하고 프로그램을 종료한다.

2. 블랙잭 게임 알고리즘

게임의 규칙에 따라 플레이어와 딜러의 승패를 결정하는 알고리즘을 구현하였다. 게임 시작과 동시에 플레이어는 1장의 카드를 받으며, 선택에 따라 추가 카드를 뽑는다. 플레이어는 21을 초과하지 않는 범위 내에서 딜러보다 높은 카드의 합을 목표로 한다. 서버(딜러)는 카드의 합계 점수를 계산하고, 자의 점수에 따라 카드를 추가로 뽑거나 멈추게 한다. 게임의 정보는 구조체를 통해 관리되며, 카드 덱은 난수 생성 함수를 사용하여 셔플된다. 소켓을 통한 통신으로 클라이언트와 서버 간에 게임 상태가 교환되며, 각 단계에서 사용자의 선택에 기반하여 게임이 진행된다.

- 랜덤화: srand와 rand 함수를 사용하여 덱 셔플링에 필요한 난수를 생성한다.
- 카드 관리: client와 server가 뽑은 카드들의 숫자 문양을 저장하는 배열, 추가로 뽑을지의 대한 상태 변수, bust 여부를 판단하는 변수, 그동안 뽑은 카드의 개수를 구조체로 묶어 관리한다.
- 게임 상태: 승패, 소지금 배팅금의 대한 변수를 선언하여 게임 상태를 관리한다.
- 배팅 시스템: 플레이어가 게임 시작 전에 배팅할 금액을 설정할 수 있다.

- 턴 관리: 게임 진행 중 플레이어의 행동(드로잉 카드, 스탠딩 등)을 관리한다.
- 게임 결과 전송: 게임 종료 후 승패 결과를 클라이언트에게 전송한다.
- 게임 반복 여부: 플레이어가 게임을 계속할지 여부를 결정할 수 있다.

3. 사용자 인터페이스

플레이어가 원할하게 게임할 수 있도록 콘솔에 디스플레이를 구현했다.

- 콘솔 클리어링: `system("clear")`를 사용하여 게임의 각 단계에서 콘솔 화면을 클리어한다.
- 표준 출력: `printf` 함수를 사용하여 사용자에게 필요한 정보를 표시한다. 게임 진행 상황, 결과, 플레이어의 잔액 등을 표시한다.
- 출력 버퍼 관리: `fflush(stdout)`를 사용하여 출력 버퍼를 관리하고, 즉시 화면에 정보를 표시한다.

4. 사용 매뉴얼

명령어	역할	설명
./server	서버 프로그램 시작	서버 프로그램을 수행하며, 수행 후 클라이언트 프로그램이 시작할 때까지 대기한다.
./client	클라이언트 프로그램 시작	클라이언트 프로그램을 수행하며, 서버가 정보를 받았다고 확인한 후 게임을 시작한다.
[number] [enter]	클라이언트 게임 진행	클라이언트 프로그램이 동작하면서, 플레이어가 자본 설정, 배팅, 게임(Hit, Stand) 플레이를 진행한다.

1) 서버 시작

1. 컴파일: gcc -o server server.c
2. 실행: ./server
 - A. 서버가 실행되면, 클라이언트가 연결을 기다린다.

2) 클라이언트 시작

1. 컴파일: gcc -o client client.c
2. 실행: ./client
 - A. 서버 IP주소와 포트 번호를 확인하고, 실행 시 서버에 자동으로 연결한다.

3) 블랙잭 게임 시작

1. 자본 입력: 플레이어가 게임에 사용할 또는 소지한 자본금을 입력한다.
2. 배팅: 이번 경기(판)에 배팅할 금액을 입력한다.
3. 카드 받기: 서버(딜러)와 플레이어 각 1장씩 카드를 받는다.
 - A. A는 1, J, Q, K는 차례로 11, 12, 13이다.
 - B. ♠ Spade 2의 경우 Spade[2], ♣ Clover 5의 경우 Clover[5], ♦ Diamond A의 경우 Dimond[1], ♥ Heart K인 경우 Heart[13]으로 콘솔창에 표현했다.
4. 플레이: 플레이어는 1. Hit(카드 뽑기), 0. Stand(카드 유지)를 선택해 플레이를 진행한다.
5. 승패 판정: 딜러는 구현된 블랙잭 알고리즘에 의해 자동으로 게임을 진행한다.
6. 결과 확인: 플레이어의 카드 합이 서버(딜러)보다 높고 21을 초과하지 않으면 승리, 서

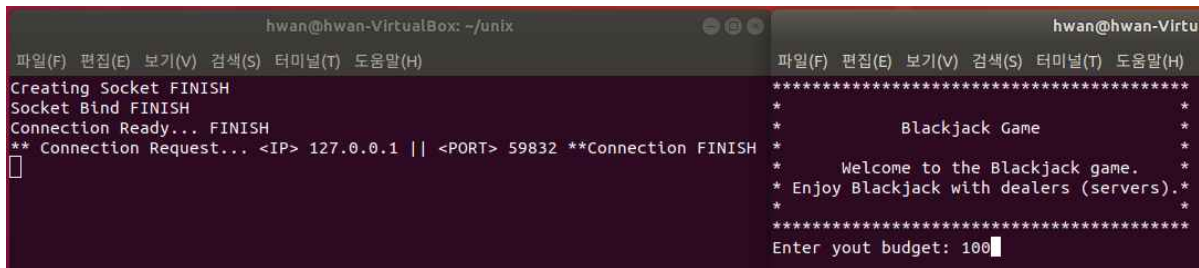
버(딜러)와 같으면 무승부, 21을 초과(Bust)하면 패배한다.

7. 추가 게임 진행 여부 확인: 게임을 추가로 진행하거나 프로그램을 종료할 수 있다.
 - A. 추가 진행 희망 시: 배팅 금액을 재 설정하고, 다시 게임을 시작한다.
 - B. 게임 종료를 원할 시: 최종 남은 자본을 출력하고 프로그램을 종료한다.

5. 사용 예 캡처

플레이 영상: <https://www.youtube.com/watch?v=5FnfmNY7cSY>

1) 클라이언트 - 서버 통신 연결 및 게임 접속



```
hwan@hwan-VirtualBox: ~/unix
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
Creating Socket FINISH
Socket Bind FINISH
Connection Ready... FINISH
** Connection Request... <IP> 127.0.0.1 || <PORT> 59832 **Connection FINISH
█

hwan@hwan-VirtualBox: ~/unix
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
*****
*
*           Blackjack Game
*
*   Welcome to the Blackjack game.
*   Enjoy Blackjack with dealers (servers).
*
*****
Enter your budget: 100█
```

- 좌 server(딜러): 서버 연결 성공
- 우client(플레이어): 게임 접속 환영 페이지 및 자본금, 배팅금 입력란 출력
- 성공적으로 연결된 것을 확인할 수 있음

2) 자본금 입력 및 배팅 금액 입력 이후 게임 진행



```
hwan@hwan-VirtualBox: ~/unix
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
*****
*
*   Player's budget is 100.
*   Betting amount for this game is 50.
*
*****
Dealer Card: Spade[11]
Player's Card: Spade[2]
█

hwan@hwan-VirtualBox: ~/unix
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
*****
*
*   Your budget is 100.
*   Betting amount for this game is 50.
*
*****
Your Card: Spade[2]
=====
||           1.Hit           ||
||           0.Stand        ||
=====
->
```

- 좌 server(딜러): 플레이어의 자본금, 본 경기 배팅 금액, 딜러와 플레이어의 카드 패 공개
- 우client(플레이어): 자신이 입력한 자본금과 현재 진행 중인 경기의 배팅 금액 출력, 자신의 패와 1. Hit(카드 뽑기), 0. Stand(카드 유지) 선택
- 현재 상황: 딜러 11점, 플레이어 2점

3) 플레이어는 Hit(카드 뽑기), Stand(카드 유지)를 하며 게임 진행

```

hwan@hwan-VirtualBox: ~/unix
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
*****
*                               *
*   Player's budget is 100.     *
*   Betting amount for this game is 50. *
*                               *
*****

Dealer Card: Spade[11] Spade[8]
Player's Card: Spade[2] Heart[2]
[]

hwan@hwan-VirtualBox: ~/unix
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
*****
*                               *
*   Your budget is 100.        *
*   Betting amount for this game is 50. *
*                               *
*****

Your Card: Spade[2] Heart[2]

=====
||               1.Hit               ||
||               0.Stand              ||
=====
->

```

- 좌 server(딜러): 딜러는 정해진 블랙잭 알고리즘에 의해 Hit을 선택했고 1장을 뽑음
- 우 client(플레이어): 플레이어는 1. Hit을 콘솔에 입력하여 한 장을 뽑음
- 현재 상황: 딜러 19점, 플레이어 4점

4) 결과 확인 (딜러 승)

```

hwan@hwan-VirtualBox: ~/unix
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
*****
*                               *
*   Player's budget is 100.     *
*   Betting amount for this game is 50. *
*                               *
*****

Dealer Card: Spade[11] Spade[8]
Player's Card: Spade[2] Heart[2]

=====
||               Game Result              ||
||               Dealer Win.              ||
=====

Waiting for player...
[]

hwan@hwan-VirtualBox: ~/unix
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
*****
*                               *
*   Your budget is 100.        *
*   Betting amount for this game is 50. *
*                               *
*****

Your Card: Spade[2] Heart[2]
Dealer Card: Spade[11] Spade[8]

=====
||               Game Result              ||
||               You Lose!               ||
=====

Your remaining budget is 50$.
Do you want to play more? (1. yes, 0. no):

```

- 좌 server(딜러): 딜러는 정해진 블랙잭 알고리즘에 의해 Stand를 선택했고 본인의 패를 유지함
- 우 client(플레이어): 플레이어는 0. Stand를 콘솔에 입력하여 본인의 패를 유지함
- 현재 상황: 딜러 19점, 플레이어 4점으로 딜러(server)가 승리가 확정됨
- 딜러와 플레이어 모두에게 결과를 제시하고, 딜러는 플레이어가 게임을 더 할지 멈출지 의사결정을 기다림

- 플레이어는 딜러가 정산해 준 자본금을 확인하고 게임을 계속하거나 멈출 수 있음

5) 결과 확인 플레이어 승

```

hwan@hwan-VirtualBox: ~/unix
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
*****
*
*      Player's budget is 100.      *
*      Betting amount for this game is 50.  *
*
*****

Dealer Card: Diamond[13] Clover[9]
Player's Card: Diamond[8] Spade[4]

=====
||      Game Result      ||
||      Dealer Lose.    ||
=====
Waiting for player...Game End.
hwan@hwan-VirtualBox:~/unix$

hwan@hwan-VirtualBox: ~/unix
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
*****
*
*      Your budget is 100.      *
*      Betting amount for this game is 50.  *
*
*****

Your Card: Diamond[8] Spade[4]
Dealer Card: Diamond[13] Clover[9]

=====
||      Game Result      ||
||      You Win!        ||
=====

Your remaining budget is 150$.
Do you want to play more? (1. yes, 0. no): 0

Finally, your remaining budget is 150.
hwan@hwan-VirtualBox:~/unix$

```

- 현재 상황: 딜러 22점, 플레이어 12점으로 딜러의 점수가 21점을 초과하여 플레이어의 승리
- 딜러와 플레이어 모두에게 결과를 제시하고, 딜러는 플레이어의 게임 연장 의사 결정을 기다림
- 플레이어는 남은 자본금을 확인하고 게임을 계속하거나 멈출 수 있음
- 0을 눌러 게임을 종료하는 경우 최종 반환 자본금을 출력함

6) 자본금을 모두 소진한 경우

```

hwan@hwan-VirtualBox: ~/unix
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
*****
*
*      Player's budget is 0.      *
*      Betting amount for this game is 100.  *
*
*****

Dealer Card: Heart[4] Diamond[9]
hwan@hwan-VirtualBox:~/unix$

hwan@hwan-VirtualBox: ~/unix
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
*****
*
*      Your budget is 100.      *
*      Betting amount for this game is 100.  *
*
*****

Your Card: Spade[13] Diamond[9]
Dealer Card: Heart[7] Clover[4]

=====
||      Game Result      ||
||      You Lose!        ||
=====

Exhausted all the budget you had
Finally, your remaining budget is 0.
hwan@hwan-VirtualBox:~/unix$

```

- 더 이상 게임을 진행할 수 없고, 안내 메시지와 함께 자동으로 게임이 종료됨