# Fejlesztői dokumentáció

Dámsa Levente

## Telepítés

Szerver : make

Kliens :make client

### Makefile tartalma

```
stackserver: stackserver.c player.c statemachine.c
    gcc stackserver.c player.c statemachine.c -o stackserver -lpthread -lm

client: client.c
    gcc client.c -o client -lm -lpthread
```

## Felépítés

A mining simulator egy tcp/ip szerver kliens program, ahol **minden kliens egy új szálban indul**. A továbbiakban bemutatom a főbb adatstruktúrákat, majd a működését a programnak.

### Adatstuktúrák

### Gép

```
typedef struct
{
    machineType type;
    int ID;                  // machine id
    double power;            // amount/tick
    double powerConsumption; // fogyaszt. (W)
    double resourceMined;
    time_t startTime; //power * elapsedtime will give the amount of resource mined.
```

```
    time_t finishTime;
} Machine;
```

A gépek működését egy vézérlési táblával oldottam meg a bővítheőget szemelőtt tartva.

Az elem struktúrával megmondja a gép következő állapotát, és a végrehajtandó függvént

```
typedef struct elem
{
    state nextState;
    void (*action)(Machine *machine);
} elem;
```

A játékban három tipusú gép van jelen, de természetesen bővíthetőek, illetve állíthatóak is a paraméterek (esetleges romlás, vagy upgrade )

## Játék

```
typedef struct Game{
    float difficulty;
    float powerCost;
    float conversionRate;
    int totalMiners;
}Game;
```

Ebben tároljuk a játék információit, az főleg **updateGame()** függvény állítja

## Játékos

```
typedef struct Player{
    float money;
    double resource;
    double powerConsumption;
    int name;
    Machine machines [MAX_MACHINES];
    state machineStates[MAX_MACHINES];
    int numMachines;
}Player;
```

Egy kliensnek az összes adatát itt tároljuk

# Machine

A machine a megadott **Power** paraméternyit fog tick-enként termelni, és a **powercost** által meghátorozott értéket fogyaszt futás közben

| Állapot/Bemenet | Idle | Busy |
|---|---|---|
| Start | Busy | Busy |
| Stop | Idle | Idle |

## Fontosabb Függvények

```
void updateGame(Game *g);
void addMachine(Player *p , machineType machineID);
void createGame(Game *g);
void tradeResource(Player *p, Game *g,double amount);
void powerBill(Player * p ,Game * g);
void addResource(Player *p, double amount);
void createGame(Game *g);
void sellMachine(Player *p,int machineIndex);
void addPlayer(Player *p,int *size,int name);
void updateForPlayer(Game *g,Player *p);
void handleMessage(Player *p,int machineIndex,eventName whathappened);
void getInfo(Player *p, Game *g);
```

- Az update függvényeket az Update thread hívja
- Gameupdate Változtatja az árfolyamot és a nehézséget a bányászók számától függően, illetve a powercost egy szinuszos hullám
- Player update a leállított gépekből kiszedi az erőforrást, hozzáadja a player erőforrásához, ellenőrzi, hogy nem ment-e csődbe.
- A kliens által kért események aszinkron módon működnek, egyből befolyásolják az állapotgépek működését
- ezen függvényeket használjuk a szerver oldalon, amikor a kliens parancsokat teljesítjük.

## Kliens szál

A kliens szál készítésekor adunk egy unique id-t, ami a **players** tömbben elhelyezett kliens indexe.

## Szerver specifikáció

TCP konkurrens szerver, Kliensenként egy új szál (pthread)

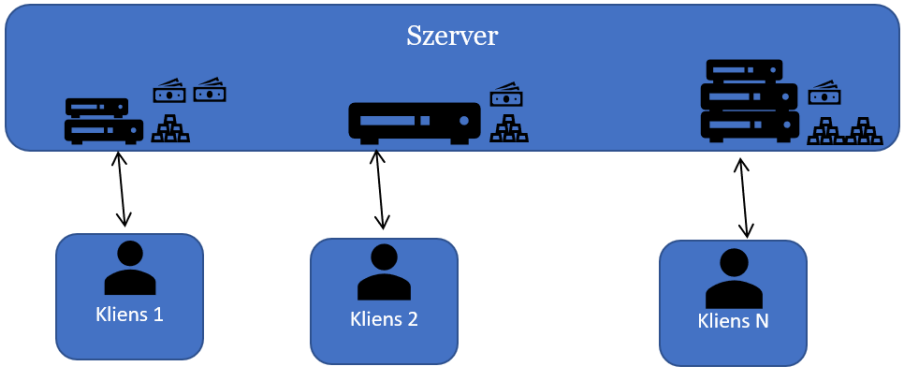


Figure 1: Architektúra

- Mindenki csak egy indexen lévő player-t ír
- Game-et kliensek csak olvassák
- Update-nél viszont mutex használata -> különben hibás adatokat olvasunk.
- Pl update-> numOfMiners <–> :getinfo command

## Kliens

### Egyszerű üzenet küldés/fogadás

- A felhasználó a standart bemeneten írhatja a meghatározott parancsokat
- A kliens fogadja, majd visszaküldi az eredményt
- Sikertelen üzenetküldésnél hibaüzenet.

### Egy updateflag 5 másodpercenként frissíti a játék paramétereit

- Küld egy ‘:getinfo’ kérést

## Jövőbeli fejlesztések

- Hozzáadni egy chat alkalmazást, amivel üzeneteket küldhetünk (jövőbeli fejlesztés, mert kell hozzá egy fogadás flag, ami mondjuk nem lenne olyan nehéz )
  - +1 bit mindig, ahol ha az 1-es, akkor jött új üzenet, és recv()-el egyet . (Vagy csak hozzáappendeli a szöveghez, úgysem számít.)