

Számítógépes Hálózatok

4. gyakorlat

PYTHON ALAPOK IV.

Socket programozás, Select

Óra eleji kisZH

- Elérés:
 - <https://canvas.elte.hu>

≡ 2018/19/1 XK85DZ-IP-08abcSZHG - Számítógépes hálózatok GY. (BSc,08,A) > Kvízek

2018/19/1

Kezdőlap

Hirdetmények

Feladatok

Fórumok

Értékelések

Résztvevők

Oldalak

Fájlok

Tematika

Tanulási eredmények

Kvízek

Modulok

Beállítások

Kvíz keresése

+ Kvíz

▼ Gyakorló kvizek

Demo kvíz

Elérhető Többes határidő | Határidő Többes határidő | 5 pont | 5 kérdés

Előző órai Feladat4

- Készítsünk egy szerver-kliens alkalmazást, ahol a kliens elküld 2 számot és egy operátort (négy alapművelet közül) a szervernek, amely kiszámolja és visszaküldi az eredményt. A kliens üzenete legyen struktúra.

Socket beállítása

- `socket.setsockopt(level, optname, value)`: az adott socket opciót állítja be
- Általunk használt *level* értékek az alábbiak lesznek:
 - `socket.IPPROTO_IP`: jelzi, hogy IP szintű beállítás
 - `socket.SOL_SOCKET`: jelzi, hogy socket API szintű beállítás
- Az *optname* a beállítandó paraméter neve, pl.:
 - `socket.SO_REUSEADDR`: a kapcsolat bontása után a port újrahasznosítása
- A *value* lehet sztring vagy egész szám:
 - Az előbbi esetén biztosítani kell a hívónak, hogy a megfelelő biteket tartalmazza (a struct segítségével)
 - A `socket.SO_REUSEADDR` esetén ha 0, akkor lesz hamis a „tulajdonság”, egyébként igaz
- Pl.: `s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)`

Select

- Több socketet is szeretnénk egy időben figyelni (a bejövő kapcsolódásokra és a meglevő kapcsolatokból való olvasásra is)
- Probléma: accept és a recv függvények blokkolnak
- Egy lehetséges megoldás lenne különböző szálak használata, de drága a szálak közti kapcsolgatás (környezetváltás, context switch)
- → A select fv. segítségével a monitorozás az op. rsz. hálózati rétegében történik
- → értesíti a programot, amikor valami olvasható a socket-ről, vagy amikor készen áll az írásra

Select

- `select.select(rlist, wlist, xlist[, timeout])`
- Az első három argumentum a „várakozó objektumok” listái:
 - *rlist*: a socketek halmaza, amelyek várakoznak, amíg készek nem lesznek az olvasásra
 - *wlist*: ... készek nem lesznek az írásra
 - *xlist*: ... egy „kivétel” nem jön
- Az opcionális *timeout* argumentum mp.-ben adja meg az időtúllépési értéket
 - (ha ez nincs megadva → addig blokkol, amíg az egyik socket kész nincs)

Select

- `select.select(rlist, wlist, xlist[, timeout])`
- Visszatér három listával:
 1. visszaadja a socketek halmazát, amelyek készek az olvasásra (adat jön)
 2. ... készek az írásra (szabad hely van a pufferükben, és lehet írni oda)
 3. ... amelyeknél egy „kivétel” jön

Select

- Az „olvasható” socketek három lehetséges esetet reprezentálhatnak:
 - Ha a socket a fő „szerver” socket, amelyiket a kapcsolatok figyelésére használunk → az „olvashatósági” feltétel azt jelenti: kész arra, hogy egy másik bejövő kapcsolatot elfogadjon
 - Ha a socket egy meglévő kapcsolat egy kientől jövő adattal → az adat a `recv()` fv. segítségével kiolvasható
 - Ha az előző, de nincs adat → a kliens szétkapcsolt, a kapcsolatot le lehet zárni

Példa hívások select-nél

- `setblocking()` vagy `settimeout()`

```
connection.setblocking(0)    # or connection.settimeout(0.0)
                             # or connection.settimeout(1.0)
connection.setblocking(1)    # or connection.settimeout(None)
```

- `select()`

```
inputs = [ server ]
outputs = [ ]
timeout=1
readable, writable, exceptional = select.select(inputs, outputs, inputs, timeout)
...
for s in readable:
    if s is server:    #new client connect
        ....
    else:
        ....          #handle client
```

Kiegészítések pythonhoz

- A **raw_input** (Python2-ben) függvény beolvas egy sort a szabványos bemenetről és visszatér ezzel sztringként.
 - Opcionálisan ennek a függvénynek lehet egy sztringet bemenetként adni, amely meg fog jelenni a képernyőn mielőtt a felhasználó elkezdené beírni az adatot.
- A **random** modulnak van a **randint(x , y)** függvénye, amely egy n számot generál véletlenszerűen úgy, hogy $x \leq n \leq y$.

Feladat 1

- Készítsünk egy TCP alkalmazást, amelyen több kliens képes egyszerre üzenetet küldeni a szervernek, amely minden üzenetre csak annyit ír vissza, hogy „OK”. (Használjuk a select függvényt!)

Órai feladat (4 pont)

- Készíts egy olyan több kliens-szerver alkalmazást, ahol a szerver gondol egy n egész számra 1 és 100 között, és a kliensek megpróbálják kitalálni! Használd a select függvényt!
 - A csatlakozott kliens-oldalon lévő felhasználó kérdéseket tesz fel: kisebb-e mint n ($< n$), nagyobb-e mint n ($> n$) vagy rákérdez egy konkrét értékre ($= n$)
 - Ezt egy "cH" formátumú struktúrával kell elküldeni
 - A szerver válaszol: "no", "yes", "win" és "end" üzenetekkel
 - Ha jó rákérdezés történik, akkor a rákérdezőnek "win" üzenet jön, a többieknek pedig "end" üzenet, és vége a játéknak (megtörténik a kapcsolat bontás)
 - Ha a szerver minden klienssel bontotta a kapcsolatot, akkor kilép

VÉGE
KÖSZÖNÖM A FIGYELMET!