Vežba 4 – Blokirajući i neblokirajući socket-i

Windows soketi mogu izvršavati ulazno-izlazne operacije u blokirajućem i neblokirajućem režimu. Razlika između ova dva režima je u tome što kada god bi operacija nad blokirajućim soketom zaustavljala izvršenja glavnog programa, neblokirajući soket ne bi dozvolio operaciju već bi vratio specijalnu "would block" poruku greške.

Pokušajmo dočarati blokirajuće i neblokirajuće režime kroz situacije iz svakodnevnog života.

Blokirajući režim

"Ne volim da putujem autobusom, jer retko kada stignem na stanicu, a da me on sačeka.

Uglavnom ja njega čekam, što ume potrajati. Jednom smo ga posle utakmice čekali par sati, ali ga nije bilo. Mora da se pokvario."

Neblokirajući režim

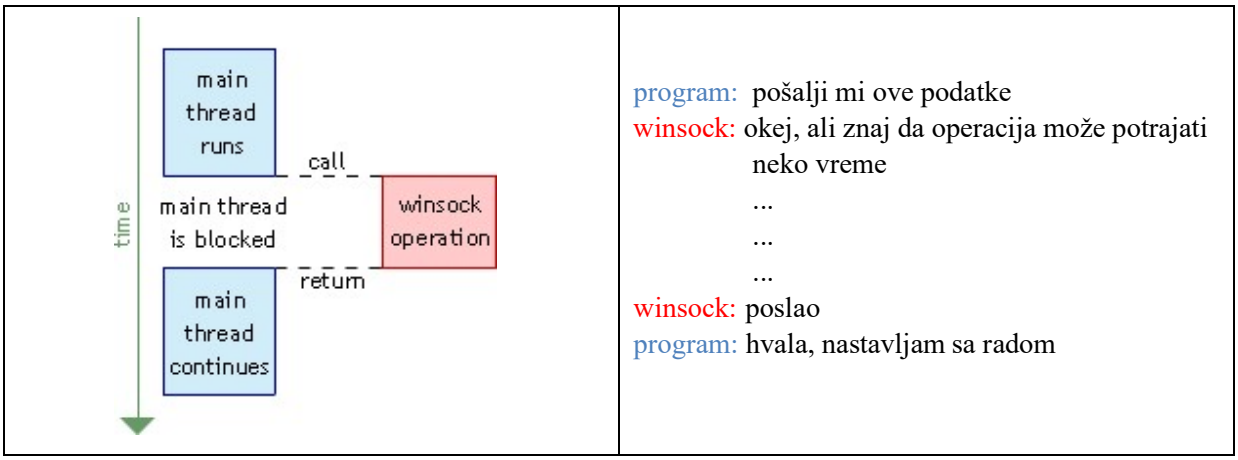
"Ne podgrevam više ništa na ringli. Dosadno mi je stojati uz šporet. Ovako stavim meso u mikrotalasnu i taman kad se završi "Državni posao", eto i moja večera javlja da je spremna."

Ista operacija pozvana u različitim trenucima može jednom blokirati izvršenje, a drugi put ne. U nastavku su date četiri tipa operacija i uslovi pod kojima će te operacije blokirati izvršenje:

- Operacije čitanja (*recv*, *recvfrom*) - primalac poruke će biti blokiran dokle god poruka koju očekuje ne pristigne sa mreže
- Operacije pisanja (*send*, *sendto*) - pošiljaoc poruke će biti blokiran ukoliko nema dovoljno slobodnog mesta u baferu za slanje poruke
- Operacija prijema konekcija (*accept*) - server će biti blokiran do trenutka prispeća novog zahteva za uspostavom konekcije
- Operacija iniciranja konekcije (*connect*) - klijent će biti blokiran do trenutka uspešnog uspostavljanja konekcije (protokol rukovanja)

1. Blokirajući soketi

Blokirajući soketi čekaju izvršenje operacije bez obzira na to da li će se ona izvršiti momentalno, posle nekog vremena ili se možda nikad neće izvršiti. Problem je, pre svega, u tome što je nit onemogućena da za vreme čekanja obavlja još neke zadatke.



Slika 1- Blokirajući soket

Jako je bitno napomenuti da se za vreme čekanja izvršenja ne troši dragoceno procesorsko vreme. Iako je nit programa blokirana, to ne sprečava druge niti da koriste procesor.

Svi soketi koje smo koristili na prethodnim vežbama bili su u blokirajućem režimu koji predstavlja podrazumevani (eng. *default*) režim soketa. U nastavku ćemo se upoznati sa neblokirajućim soketima, čija implementacija je zahtevnija u odnosu na blokirajuće sokete, ali omogućava znatno bolje performanse komunikacije.

2. Neblokirajući soketi

Funkcije koje se pozivaju nad soketom koji je postavljen u neblokirajućem režimu nikada neće blokirati izvršenje programa, već će povratak iz funkcije uslediti odmah nakon poziva. Neblokirajuće operacije nad soketom će se izvršiti, dok će blokirajuće generisati grešku koja će ukazati da nije bilo dovoljno vremena da se operacija izvrši.

Soket se može postaviti u neblokirajući režim pomoću funkcije **ioctlsocket** koristeći za argument **FIONBIO** komandu. Ova funkcija može se koristiti za sve tipove soketa, nezavisno od stanja u kome se soketi nalaze. Takođe, neke funkcije (poput funkcija *WSAEventSelect* i *WSAAsyncSelect)*

mogu implicitno postaviti režim soketa na neblokirajući. Ove napredne mogućnosti nećemo koristiti u toku ovog kursa.

Deklaracija `ioctlsocket` funkcije, njen detaljan opis i primer su dati u nastavku:

```
int ioctlsocket(SOCKET sock,long command,unsigned long *mode);
```

Funkcija:	Opis:
ioctlsocket	Omogućava promenu režima blokiranja soketa.
Parametri:	Opis:
SOCKET sock[in]	Soket kome želimo promeniti režim blokiranja
long command[in][out]	Za odabir komande blokiranja potrebno je upisati FIONBIO (sem komande blokiranja, postoji još korisnih komandi, ali one neće biti tema ovog kursa)
unsigned long *mode [in]	Pokazuje na promenljivu u kojoj se navodi režim blokiranja koji želimo postaviti (0 - blokirajući režim, 1 - neblokirajući režim)
Povratna vrednost:	Opis:
int	Ako se funkcija uspešno izvrši vraća vrednost 0. U suprotnom, vraća se SOCKET_ERROR, a kod greške se dobija nakon poziva funkcije WSAGetLastError.

U nastavku je dat primer u kome se koristi neblokirajuci režim rada TCP soketa.

```
//Postavljanje soketa u neblokirajuci rezim
unsigned long mode = 1; //non-blocking mode
iResult = ioctlsocket(acceptedSocket, FIONBIO, &mode);
if (iResult != NO_ERROR)
	printf("ioctlsocket failed with error: %ld\n", iResult);

// Poziv operacije nad soketom
iResult = recv(acceptedSocket, dataBuffer, BUFFER_SIZE, 0);

//Provera da li se operacije uspesno izvrsila
if (iResult != SOCKET_ERROR) {
	// Neblokirajuca operacija koja je uspesno izvrsena
}
else
{
	if (WSAGetLastError() == WSAEWOULDBLOCK) {
		// U pitanju je blokirajuca operacija koja zbog rezima
		// soketa nece biti izvrsena
	}
	else {

		// Desila se neka druga greska prilikom poziva operacije
	}
}
```

Na početku potrebno je postaviti soket u neblokirajuci režim pomoću funkcije *ioctlsocket*. Nakon toga poziva se operacija nad soketom *recv*. Pomoću rezultata koji je vratila funckija *recv* možemo protumačiti da li se funkcija uspešno izvršila. U slučaju da nije, potrebno je pozvati funkciju *WSAGetLastError* kako bi se doznao kod greške. Ukoliko je kod greške *WSAEWOULDBLOCK* to znači da je operacija nad soketom zahtevala blokiranje, što joj zbog režima soketa nije bilo dozvoljeno.

Polling model

Pooling model podrazumeva ponavljanje poziva operacije nad neblokirajućim soketom, dokle god se operacija ne izvrši uspešno.

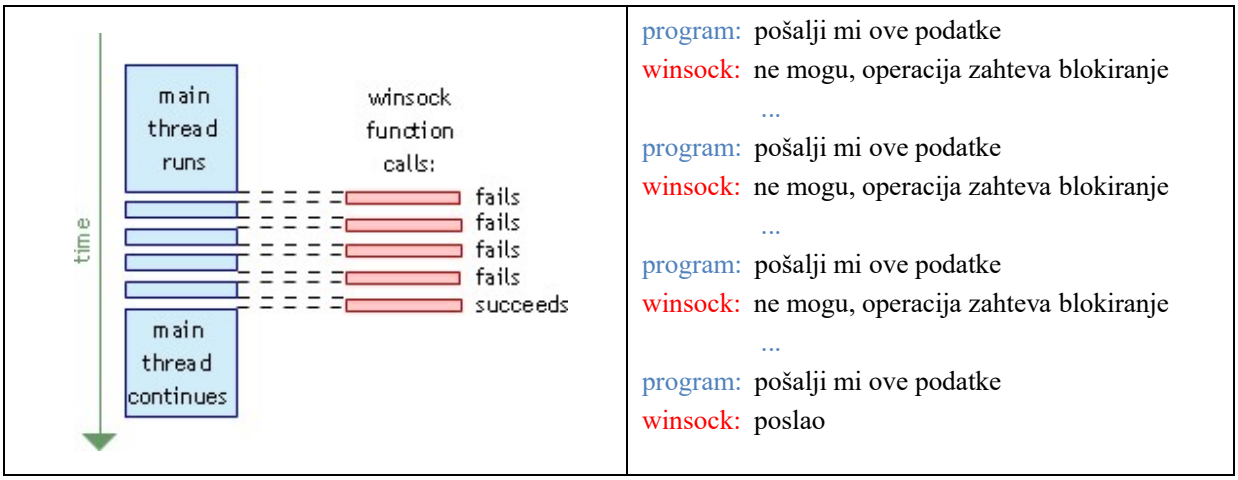


Figure 2 – Pooling model

U odnosu na blokirajući režim, ovaj model neblokiranja nudi ipak neku vrstu kontrole u okviru petlje koja omogućava da se za vreme čekanja izvrše još neke druge operacije ili da se prekine čekanje kada se na primer nekoj promenljivoj promeni vrednost. S druge strane, za vreme čekanja se troši dragoceno procesorsko vreme i glavni program je za vreme čekanja blokiran.

Zadatak 1

Koristeći primer implementacije blokirajućeg UDP klijenta i servera koji je dat u prilogu materijala potrebno je implementaciju server unaprediti sledećim mogućnostima.

1. Omogućiti prijem svih poruka čije odredište je port 15001.
2. Implementirati pooling model neblokiranja na serveru, koji omogućava da se prilikom prijema poruka od klijenta ne blokira nit procesora.
    - Za postavljanje neblokirajućeg režima koristiti funkciju ioctlsocket.
    - Za realizaciju iterativnog pokušaja prijema poruka sa mreže koristiti programsku petlju.
3. Smanjiti potrošnju procesorskog vremena za vreme čekanja prispeća podataka.
    - Između dva pokušaja potrebno je sačekati 1 sekundu. Za čekanje koristiti funkciju čija deklaracija je data u nastavku:
      `Sleep(unsigned int miliseconds);`
4. Za vreme čekanja potrebno je ispisati na ekran broj trenutnog pokušaja.
5. Obezbediti mogućnost prekida prijema poruka na serveru ukoliko u periodu od 30 sekundi ne primi nijednu poruku.

Zadatak 2

Koristeći primer implementacije UDP klijenta i servera koji je dat u prilogu materijala za vežbu, potrebno je omogućiti na serveru prijem poruka u neblokirajućem režimu.

1. Na serveru napraviti dve UDP utičnice i povezati ih sa adresama i portovima koji su dati u nastavku:
    - serverSocket1
        - IP adresa: 127.0.0.1
        - Port: 17010
    - serverSocket2
        - IP adresa: sve dostupne adrese
        - Port: 17011
2. Implementirati **polling model** neblokiranja na serveru koji će omogućiti da se prilikom prijema poruka od klijenata ne blokira nit procesora.
    a) Obezbediti neblokirajući režim izvršavanja operacija nad serverskim utičnicama.
    b) Smanjiti potrošnju procesorskog vremena za vreme čekanja prispeća poruka. Sačekati do narednog pokušaja prijema poruke 1,5 s. Za ovu namenu koristiti funkciju `Sleep(unsigned int miliseconds);`
    c) Kada poruka stigne, ispisati sadržaj poruke i koja utičnica ju je primila.
3. Prepoznati koji klijent je prvi poslao poruku pa njemu dodeliti 1 poen. Napomena: pri testiranju programa slati naizmenično poruku sa jednog, pa drugog klijenta proizvoljnim redosledom.
4. Na klijentskoj strani potrebno je na početku programa učitati iz komandne linije port servera ka kome će se slati poruke. To će nam omogućiti da se izvršavanjem istog klijentskog koda mogu slati poruke ka različitim serverskim portovima (portovi 17010 i

17011).

5. Omogućiti klijentu da može poslati više poruka serveru. Ukoliko klijent upiše "kraj" potrebno je ugasiti klijenta.

Zadatak 3

Koristeći primer implementacije TCP klijenta i servera koji je dat u prilogu materijala za vežbu, potrebno je omogućiti sledeće funkcionalnosti:

1. Server koji aktivno sluša zahteve za uspostavom veze na portu 18010 treba da uspostavi konekciju sa 2 klijenta. Prilikom uspostave veze ispisati adresne informacije o klijentu sa kojim je veza ostvarena.

Napomena: komunikacija sa klijentima će se odvijati tek nakon što se oba klijenta konektuju.

2. Implementirati **polling model** neblokiranja na serveru koji će omogućiti da se prilikom prijema poruka od klijenata ne blokira nit procesora.
   a) Obezbediti neblokirajući režim izvršavanja operacija nad utičnicama namenjenim razmeni podataka sa klijentima.
   b) Smanjiti potrošnju procesorskog vremena za vreme čekanja prispeća poruka. Sačekati do narednog pokušaja prijema poruke 1,5 s. Za ovu namenu koristiti funkciju `Sleep(unsigned int miliseconds);`
   c) Kada poruka stigne, ispisati sadržaj poruke i koja utičnica ju je primila.

3. Omogućiti klijentu da može poslati više poruka serveru. Ukoliko se na klijentu unese "exit" potrebno je ugasiti klijenta.

4. Omogućiti da klijent u okviru jedne poruke pošalje niz od 3 celobrojne vrednosti (izabrati da li će klijent slati vrednosti tipa int ili short).

5. Na serveru nakon prijema poruke omogućiti pravilan ispis niza od 3 broja.

Server kao odgovor na klijentovu poruku šalje poruku koja sadrži zbir vrednosti koje muje klijent prosledio u nizu brojeva. Na klijentu ispisati primljeni odgovor.

Zadatak 4

Koristeći primer implementacije TCP klijenta i servera koji je dat u prilogu materijala za vežbu, potrebno je omogućiti sledeće funkcionalnosti.

1. Na TCP serveru obezbediti da se korišćenjem **polling modela neblokiranja prihvate konekcije sa 2 klijenta**:
   a) Obezbediti neblokirajući režim izvršavanja operacija nad serverskom utičnicom koja sluša zahteve za uspostavom veze na portu 19010.
   b) Smanjiti potrošnju procesorskog vremena za vreme čekanja zahteva. Do narednog pokušaja prijema konekcije sačekati 2 s. Za ovu namenu koristiti funkciju `Sleep(unsigned int miliseconds);`
   c) Kada zahtev za vezom stigne, prihvatiti ga i ispisati adresne informacije o klijentu sa kojim je uspostavljena veza.

Napomena: komunikacija sa klijentima će se odvijati tek nakon što se oba klijentakonektuju. Zadatak je moguće uraditi tako što će se ovaj korak pojednostaviti prijemom konekcija u podrazumevanom (blokirajućem) režimu.

2. Implementirati **polling model** neblokiranja na serveru koji će omogućiti da se prilikom **prijema poruka od klijenata** ne blokira nit procesora.
   a) Obezbediti neblokirajući režim izvršavanja operacija nad serverskim utičnicama namenjenim klijentima.
   b) Smanjiti potrošnju procesorskog vremena za vreme čekanja prispeća poruka. Sačekati do narednog pokušaja prijema poruke 1 s. Za ovu namenu koristiti funkciju `Sleep(unsigned int miliseconds);`
   c) Kada poruka stigne, ispisati sadržaj poruke i koja utičnica ju je primila.

3. Omogućiti klijentu da može poslati više poruka serveru. Ukoliko se na klijentu unese "exit" potrebno je ugasiti klijenta.

4. Omogućiti da klijent u okviru jedne poruke pošalje niz od 3 celobrojne vrednosti tipa int.

5. Na serveru omogućiti adekvatan ispis primljenih celobrojnih vrednosti iz niza.

6. Server kao odgovor klijentu šalje najveću vrednost iz primljenog niza u formatu: "Najveći poslati broj je 100." u primeru gde je sever primio niz vrednosti 50, 99 i 100.