

Operativni sistemi i RuO

Vježbe 5

(dopuna materijala)

Plan vježbe

- Biblioteke – **threads i atomic**
 - **Multiprogramiranje** - upravljanje sa više procesa u jednoprocorskom sistemu
 - **Multiprocesiranje** - upravljanje sa više procesa unutar multiprocorsa
- **Semafori** – mehanizmi u OS-u za razmjenu signala između procesa.
- **Mutexi** – *Binarni semafori (0 ili 1)*
- Linux za **systemsku administraciju**
- **Primjeri**
- **Prezentacija zadaća**

<threads> i <atomic>

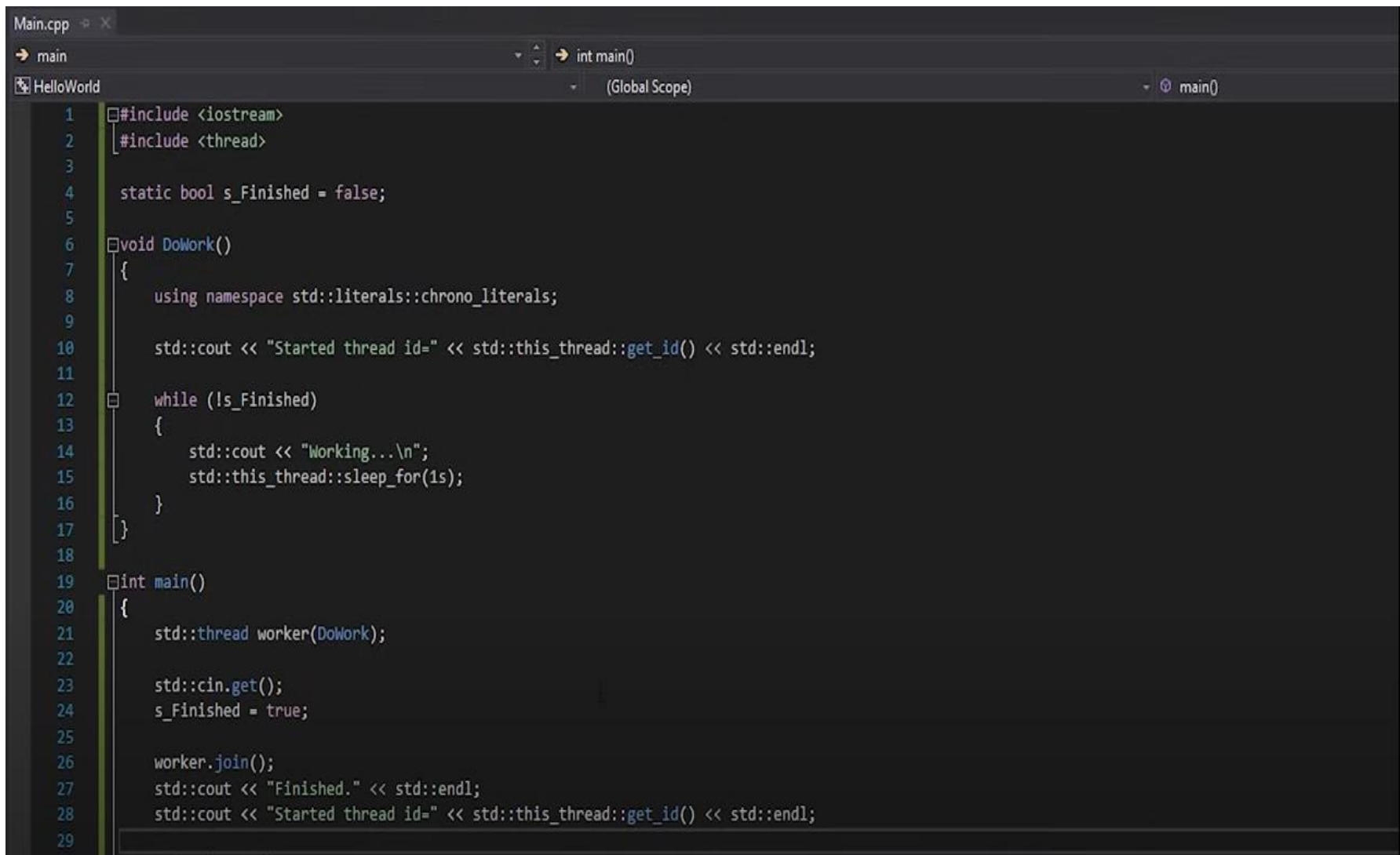
- Biblioteka uključena je u standardnu C++ biblioteku i koristi se za rad s nitima.
- Niti su korisni kada želimo da se dio koda izvršava u pozadini, bez blokiranja glavne niti. Niti se mogu koristiti za izvršavanje dugotrajnih zadataka
- Primjer učitavanje velikih datoteka ili obradu velikih količina podataka.

<atomic>

- `include <atomic>`
- Biblioteka u C++ omogućava rad s atomičnim tipovima podataka.
- Atomski tipovi podataka su tipovi podataka koji se mogu čitati i pisati u jednoj operaciji
- Navedeno osigurava da se „**niti**“ neće natjecati za pristup zajedničkim varijablama.

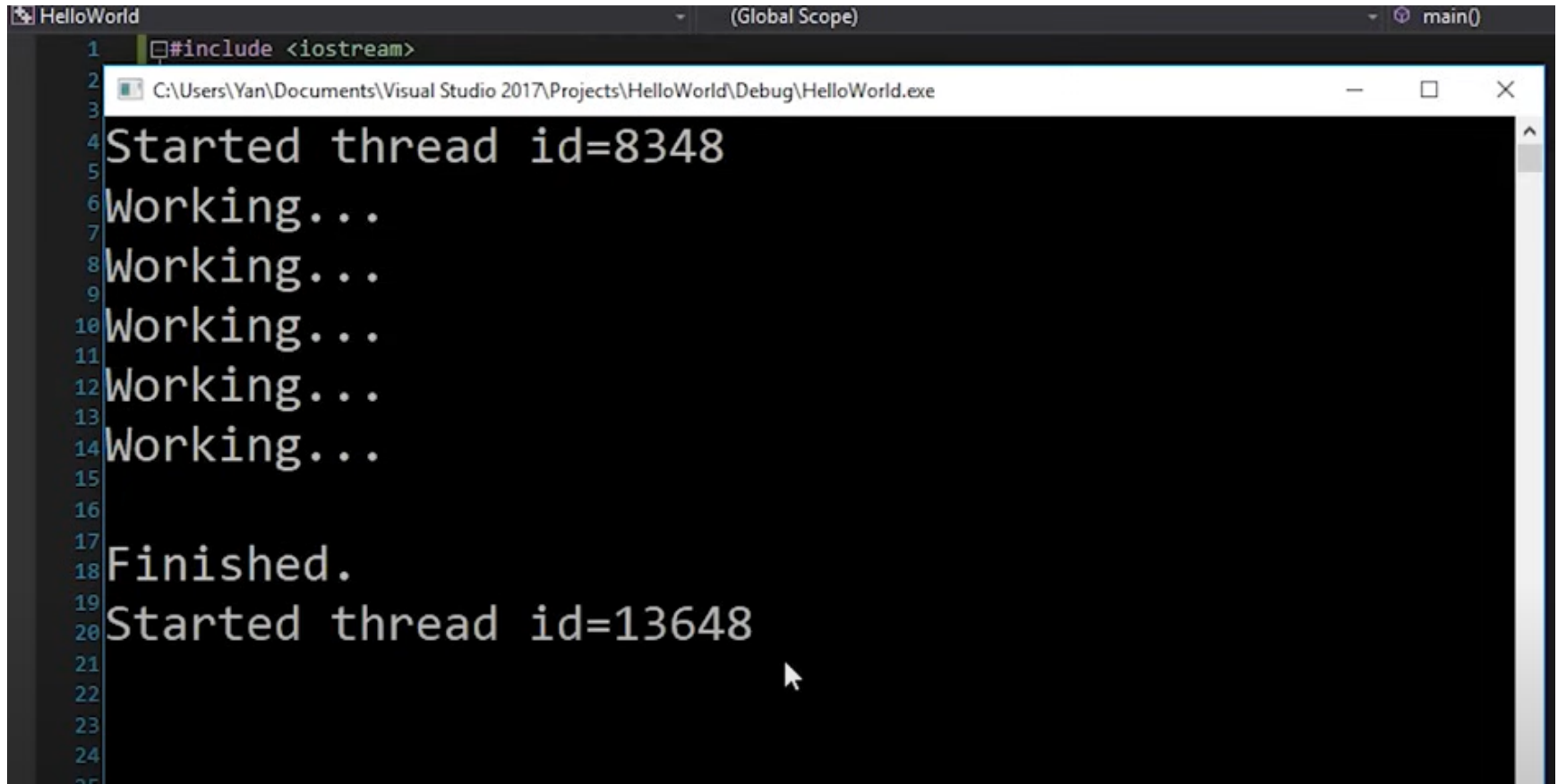
Primjer kako C++ omogućava <threading>

Prikaz petlje tekstualne animacije „Učitavanje“ u C++-u



```
1  #include <iostream>
2  #include <thread>
3
4  static bool s_Finished = false;
5
6  void DoWork()
7  {
8      using namespace std::literals::chrono_literals;
9
10     std::cout << "Started thread id=" << std::this_thread::get_id() << std::endl;
11
12     while (!s_Finished)
13     {
14         std::cout << "Working...\n";
15         std::this_thread::sleep_for(1s);
16     }
17 }
18
19 int main()
20 {
21     std::thread worker(DoWork);
22
23     std::cin.get();
24     s_Finished = true;
25
26     worker.join();
27     std::cout << "Finished." << std::endl;
28     std::cout << "Started thread id=" << std::this_thread::get_id() << std::endl;
29 }
```

Debug mode- rezultat petlje



```
1 #include <iostream>
2
3
4 Started thread id=8348
5
6 Working...
7
8 Working...
9
10 Working...
11
12 Working...
13
14 Working...
15
16
17 Finished.
18
19 Started thread id=13648
20
21
22
23
24
25
```

Visual Studio 2017 - Okruženje

Semafori

- Semafori su alat koji se koristi u operativnim sistemima i paralelnom programiranju kako bi se sinkronizirali procesi i threadovi.
- Semafor je vrsta varijable koja se koristi za upravljanje pristupom zajedničkim resursima.
- Semafor drži broj dozvola, koje threadovi moraju dobiti prije nego što mogu pristupiti zajedničkom resursu.
- Opšti i binarni semafor
- Semafori su jednostavan mehanizam OS-a za razmenu signala između procesa.
- Da bi se poslao signal, izvršava se **SemSignal(s)**, a da bi se primio **SemWait(s)**.
- Opšti semafor se može inicijalizovati na pozitivnu vrednost
 - semWait() umanjuje vrednost semafora. Ako postane negativna, proces se blokira. Ako ostane pozitivna, nastavlja se dalje.
 - semSignalB() uvećava vrednost semafora. Ako postane ≤ 0 , vrši se deblokada procesa koji je blokiran pomoću semWait()

Semaphore struktura sa dvije metode, wait() i signal()

```
#include <iostream>
#include <thread>
#include <atomic>
#include <vector>

struct Semaphore {
    std::atomic<int> count;

    Semaphore(int initial_count = 1) : count(initial_count) {}

    void wait() {
        int expected = 1;
        while (!count.compare_exchange_strong(expected, 0))
        {
            expected = 1;
            std::this_thread::yield();
        }
    }

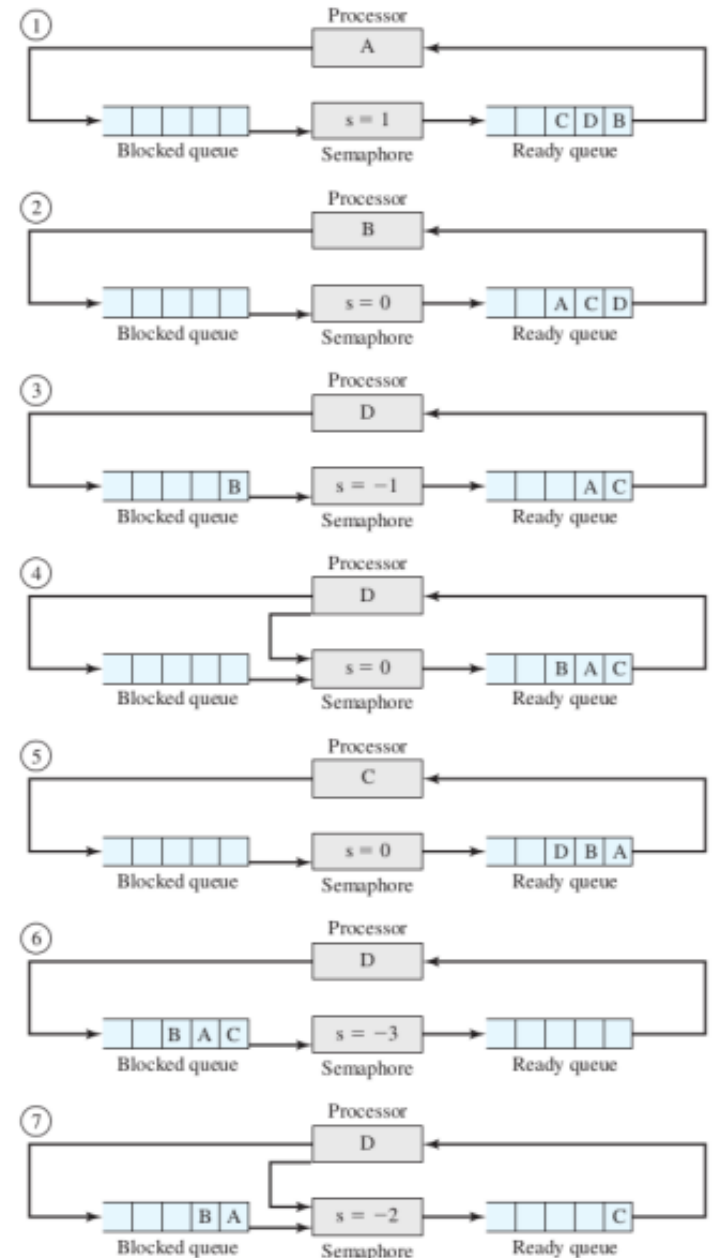
    void signal() {
        count.store(1);
    }
};

locked.store(false, std::memory_order_release);
};
```

***U primjeru se koristi *Semaphore struktura* koja ima dva metoda, wait() i signal(), koje čekaju i signaliziraju semafor. wait() metoda smanjuje vrijednost brojača semafora i čeka dok se ne poveća na 1, a signal() metoda povećava vrijednost brojača za 1.

Primjer korištenja semafora

- Procesi **A**, **B** i **C** čekaju podatke koje obezbjeđuje proces **D**



Mutexi

- **Binarni semafori** – mutexi
- Klasična implementacija mutexa u OS-u je nastala kao rezultat rješavanja „**race condition**“ u kritičnim sekcijama procesa. *Race condition – stanje utrke.*
- Semafor se može inicijalizovati na 0 ili 1
- **semWaitB()** provjerava vrijednost semafora. ako je 0, blokira se proces koji je pozvao semWaitB(). Ako je 1, mijenja se na 0 i nastavlja se dalje.
- **semSignalB()** provjerava da li je neki proces na datom semaforu blokiran. Ako jeste, deblokira se. Ako nema blokiranih procesa, vrijednost semafora se postavlja na 1.

Struktura opšte i binarnog semafora

Opšti

```
struct semaphore {
    int count;
    queueType queue;
};

void semWait(semaphore s)
{
    s.count--;
    if (s.count < 0) {
        /* place this process in s.queue */;
        /* block this process */;
    }
}

void semSignal(semaphore s)
{
    s.count++;
    if (s.count <= 0) {
        /* remove a process P from s.queue */;
        /* place process P on ready list */;
    }
}
```

Binarni

```
struct binary_semaphore {
    enum {zero, one} value;
    queueType queue;
};

void semWaitB(binary_semaphore s)
{
    if (s.value == one)
        s.value = zero;
    else {
        /* place this process in s.queue */;
        /* block this process */;
    }
}

void semSignalB(binary_semaphore s)
{
    if (s.queue is empty())
        s.value = one;
    else {
        /* remove a process P from s.queue */;
        /* place process P on ready list */;
    }
}
```

Linux za sistemsku administraciju

- 1. Sistemski Logovi*
- 2. Komande za manipulaciju procesa.*
- 3. Komande za procese*
- 4. Administracija grupa i korisnika*
- 5. Varijable okruženja*
- 6. Monitoranje (nadzor) korisnika*

Sistemske logove

- Sistemske Logove se nalaze na putanji „**/var/log/**“.
- Čitanje logova u Linuxu je važan postupak za praćenje različitih događaja i problema na sistemu.
- Postoje različiti logovi koje Linux sustav generiše, a neki od najčešćih su sljedeći:
 - **Syslog** - glavni dnevnik događaja na sistemu koji sadrži sve važne događaje vezane uz sistem, uključujući kernel poruke, poruke aplikacija, poruke sistema sigurnosti. • **Auth.log** - dnevnik koji sadrži informacije o prijavi i autentikaciji korisnika na sistemu, uključujući i neuspjele prijave.
 - **Apache access i error logovi** - dnevnik koji sadrži informacije o pristupu web browseru, uključujući korisničke zahtjeve i greške.
 - **Mail.log** - dnevnik koji sadrži informacije o e-pošti, uključujući slanje, primanje i greške.
 - **Cron.log** - dnevnik koji sadrži informacije o agendama na sistemu, uključujući i pokretanje skripti i programskih alata na rasporedu.

Za čitanje logova "tail", koja prikazuje zadnjih nekoliko linija dnevnika u Linuxu.

Primjer, "tail -f /var/log/syslog" prikazat će zadnjih 10 linija **syslog** datoteke i nastaviti s prikazivanjem novih linija koje se dodaju u dnevnik dok se ne prekine naredba (CTRL+C).

Komande za održavanje sistema

- Shutdown
- Reboot
- Halt
- Init

Komande za manipulaciju procesa

- Systemctl
- Ps
- Top
- Kill Page

Administracija grupa i korisnika

- Useradd
- Groupadd
- Userdel
- Groupdel
- Usermod

Specifčne lokacije

1. /etc/passwd
2. /etc/group
3. /etc/shadow

Varijable okruženja

- **Environment variables** (varijable okruženja) su vrijednosti koje su dostupne u okruženju operativnog sistema i koriste se za razne potrebe aplikacija i sistema.
- One sadrže podatke kao što su putanja do datoteka, korisnička imena, postavke sistema, i drugo.
- U Linuxu, neke od uobičajenih okružnih varijabli uključuju:
 - HOME - putanja do mape matičnog direktorija trenutno prijavljenog korisnika.
 - PATH - popis mapa u kojima operativni sustav traži izvršne datoteke
 - USER - korisničko ime trenutno prijavljenog korisnika.
 - SHELL - putanja do ljuske koju koristi trenutni korisnik.
 - LANG - postavka jezika sustava. Naredba "env" prikazuje sve okružne varijable na vašem sistemu.
 - Primjer: "env | grep USER" će izlistati sve varijable okruženja koje sadrže riječ "USER" u njihovom imenu. Također, naredba "export" da bi se postavila vrijednost okružne varijable. Primjer: "export MY_VAR=my_value" će postaviti okružnu varijablu "MY_VAR" na vrijednost "my_value". Ova varijabla će biti dostupna u svim sljedećim procesima koji se pokrenu u trenutnom okruženju. Lokacija ovih varijabli se nalazi u /etc/.bashrc

Linkovi i tutorijali za učenje

- Google Classroom – sekcija Vježbe/Informacije A2 grupa

Diskusija

- *Ponavljjanje gradiva sa prethodnih vježbi... !*
- *Vježbanje- linux komande - terminal*
- *Prezentacija zadaća (zadatak 2) !*
- *Upiti/konsultacije i ostalo....*

Hvala na pažnji

