4 – Лаборатория иши

Тезкор мурожаат шарти таҳдиди (Race condition)

Ишдан мақсад: C++ дастурлаш тилида Тезкор мурожаат шарти таҳдиди билан танишиб чиқиш.

Назарий қисм

(Race condition ёки race hazard). Бу дастурий маҳсулотнинг ёки электрон тизимнинг ўзини тутиш ҳолати бўлиб, чиқиш қиймати бошқариб бўлмас бошқа ҳодисалар кетма-кетлиги ёки вақтига боғлиқ бўлади. Дастурлашда бу ҳолда хатолик юзага келиб, иккита сигнал биринчи чиқиш учун ҳаракат қилади. Бу ҳодиса асосан, дастурлашда параллел ҳисоблашда (thread) юзага келади.

4.1 – расм. Тезкор мурожаатда ҳаётий мисол

Уч турдаги мустақил оқимлар бўлиши мумкин:

- Оқимлар (thread);
- Жараёнлар;
- Вазифалар.

Ушбу ҳодисанинг юзага келиши учун уч турдаги хусусият мавжуд:

- *Конкуренциянинг мавжудлиги.* Камида иккита оқим амалга оширилган бўлиши шарт;
- *Тақсимланган объект.* Тақсимланган объектлар барча конкурент оқимлар томонидан бошқарилиши шарт;
- *Ҳолатни ўзгариши.* Камида битта оқим тезкор мурожаатни амалаг ошириши шарт.

Қуйида иккита оқим томонидан ўз қийматини бирга ошириш учун бажарган тезкор мурожати келтирилган. Агар тизим тўғри ишлаганда қуйидаги натижа олиниши шарт эди.

Thread 1	**Thread 2**		**Integer value**
			0
read value		←	0
increase value			0
write back		→	1
	read value	←	1
	increase value		1
	write back	→	2

Аммо, тезкор мурожаат натижасида қуйидаги ҳолат келиб чиқди:

Thread 1	**Thread 2**		**Integer value**
			0
read value		←	0
	read value	←	0
increase value			0

	increase value		0
write back		→	1
	write back	→	1

Бу таҳдид мавжуд дастурий маҳсулотларда time-of-check-to-time-of-use (TOCTTOU) заифлиги мавжуд бўлади.

Муаммо бирор оқим **“check-then-act”** ни амалга ошириш вақтида (масалан, "check" агар X бирор катталикга тенг бўлса, у ҳолда "act" X билан боғлиқ бирор амал) ва бошқа бир оқим бу қийматга "check" ва "act" оралиғига таъсир қилади. Масалан:

```
if (x == 5) //"Check"
{
   y = x * 2; //"Act"
   // агар бошқа бирор оқим x ни "if (x == 5)" ва "y = x * 2" оралиғида ўзгартирса
   // y катталик 10 га тенг бўлмайди.
}
```

Одатда бу таҳдидларни олдини олишда дастурлаш тилларида глобал ўзгарувчини қулфлаб қўйиш усулларидан фойдаланилади.

```
// x учун қулфлашни амалга ошириш
if (x == 5)
{
   y = x * 2; // қулфлаш олиб ташлагмагунга қадар бошқа оқим томонидан ўзгартириб бўлмайди
              // Шунинг учун y = 10
}
// x ни қулфдан озод этиш
```

Амалий қисм

Қуйида оқимдан фойдаланган ҳолда ва асосий функция ичида чоп этиш буйруғидан фойдаланилиб, қийматлар чоп этилган код берилган:

```
#include <iostream>
#include <thread>
void thread_function()
{
    for (int i = -100; i < 0; i++)
        std::cout << "thread function: " << i <<
"\n";
}
int main()
{
    std::thread t(&thread_function);
    for (int i = 0; i < 100; i++)
        std::cout << "main thread: " << i << "\n";
    t.join();
    return 0;
}
```

Натижадан кўриниб турибдики, бу ҳолда тартиб бузилган.

Бу ҳолатни олдини олиш учун C++ дастурлаш тилида `mutex`: функциясидан фойдаланилади ва у билан чоп этиш буйруғи қулфланади:

```
#include <iostream>
#include <thread>
#include <mutex>
#include <string>
std::mutex mu;
void shared_cout(std::string msg, int id)
{
    mu.lock();
    std::cout << msg << ":" << id << std::endl;
    mu.unlock();
}
```

```
void thread_function()
{
    for (int i = -100; i < 0; i++)
        shared_cout("thread function",i);
}
int main()
{
    std::thread t(&thread_function);
    for (int i = 0; i < 100; i++)
        shared_cout("main thread", i);
    t.join();
    return 0;
}
```

Назорат саволлари

1. C++ да оқимларни ҳосил қилиш усуллари.
2. Race conditiondан қандай ҳимояланиш мумкин (C ва C++).
3. Ҳар бир қарши ҳимоя усулини мисоллар орқали тушунтиринг.
4. Deadlock нима ва у қайси вақтда келиб чиқади.

Изоҳ: ҳар бир назорат саволларига ёзма жавоб берилсин.