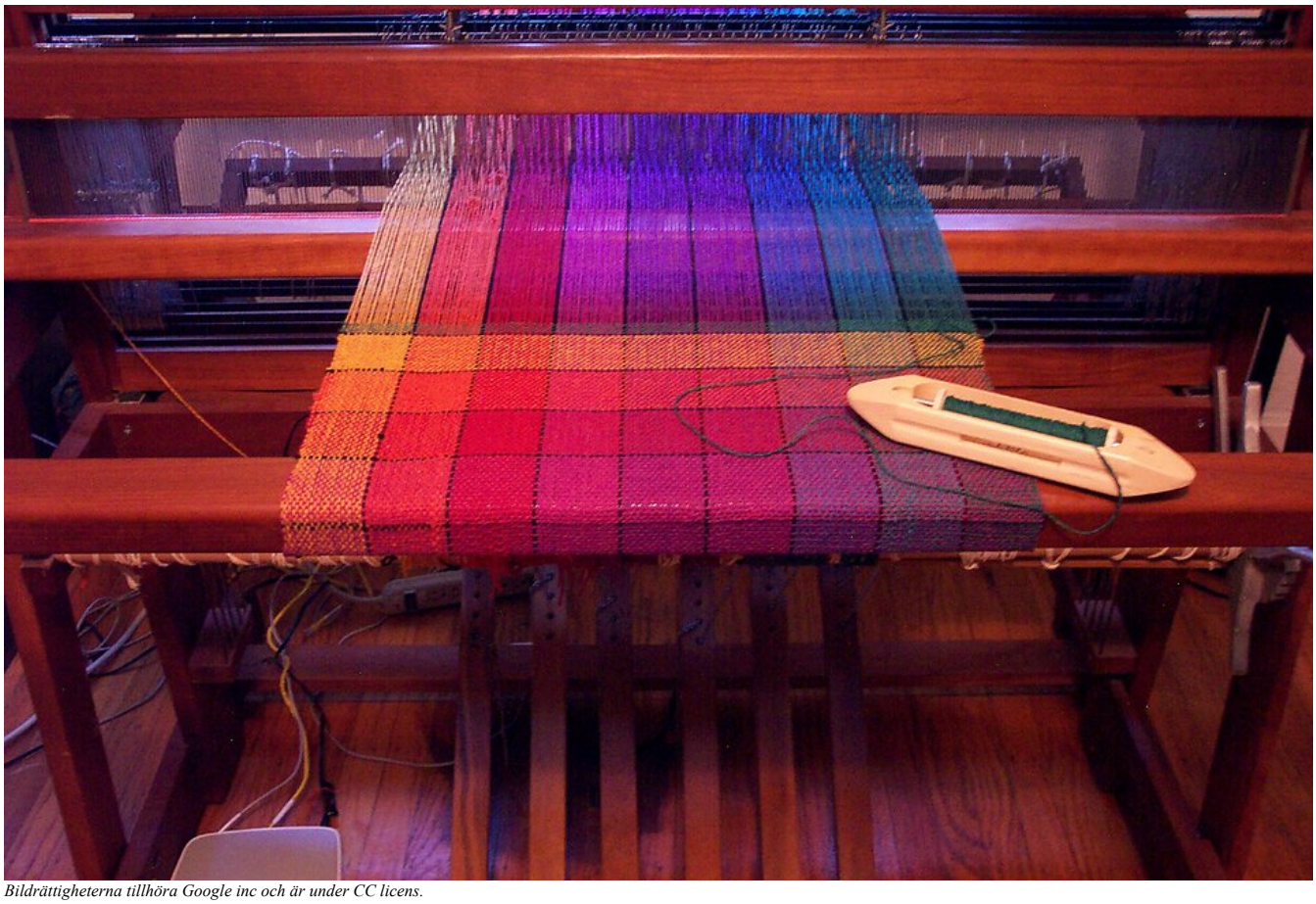
Systemnära programmering 5DV088HT19

Obligatorisk uppgift 4 MFIND



Bildrättigheterna tillhöra Google inc och är under CC licens.

Max Malmer(c17mmr)
Kontakt: c17mmr@cs.umu.se
Kursansvarig: Mikael Ränner

2019-10-25, v.1.0

Innehållsförteckning

1. Förord

Till att börja med så ska jag direkt säga att jag inte är supernöjd med min lösning. Eftersom att jag inte fick #include <semaphore.h> att fungera så använde jag mig i slutändan utav semops:

```
#include <sys/ipc.h>
#include <sys/sem.h>
```

Semops verkar fungera relativt bra, men jag kan inte påstå exakt att jag vet precis vad som händer. Jag tror också att det kan vara så att jag ställer semanphoren för lågt vid tillfällen. Den Dokumentation som jag utgått från är specat i referenserna. Utifrån ståndpunkten att mitt program kanske inte är så vettigt byggt så är jag verkligen mer än okej med ett O, kom gärna med kommentarer på där ni tror jag gjort någon tankevurpa. För jag misstänker att jag gjort flera. Jag tror också att jag kanske har lite många globala variabler samt att jag har en strukt kvar i programmet som sedan blev onödig. Minneshanteringen vart lite mysko också, det finns inga läckor men programmets läsbarhet hade tjänat på att på att ha en lista där jag kunnat skicka in en funktion. Det fick jag dock inte att fungera.

2. Trådsäkerhet

Programmet som det är nu bör vara trådsäkert. Jag använder mig av en kombination av mutex lås och semanphorer för att garantera detta. Om programmet körs med Helgrind så får man dock output på ett misstänkt race i memcpy i printf. Detta borde dock inte vara fallet eftersom jag uppdaterat med följande :

```
#define _POSIX_C_SOURCE 200809L
#define _XOPEN_SOURCE
```

Som jag förstått så ska printf därgenom vara trådsäkert sedan kring år 2000.

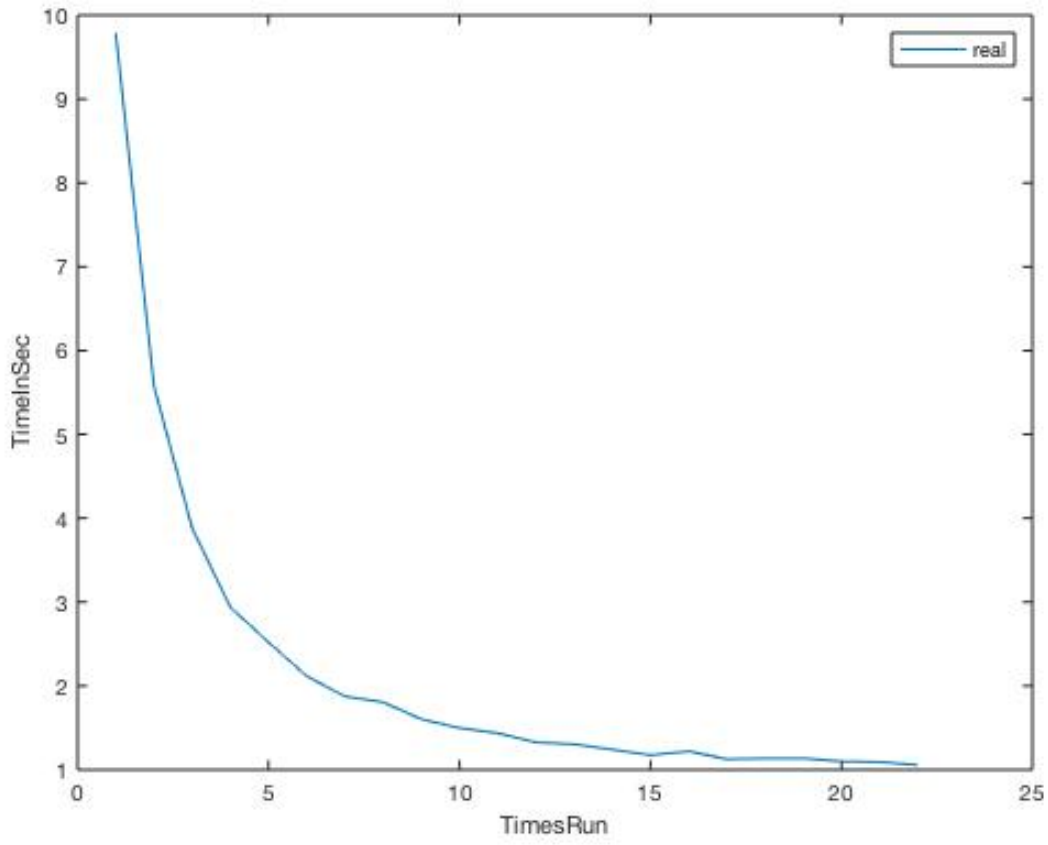
Något som dock kan vara fallet är att jag kanske låser programmet eller håller mina trådar med semanphorer lite för ofta. En tanke är t.ex. att jag låser upp semanphorer både när en ny path läggs i kön, men också när en tråd lämnar trådfunktionen. I min pseudokod så tänkte jag att det skulle vara möjligt att inte behöva låsa upp semanphoren där men så verkar inte vara fallet. Så jag misstänker att det kan vara något som jag glömt. Dock så verkar detta inte ställa till problem eftersom arbetsbelastningen (utan valgrind) alltid är jämn.

3. Arbetsfördelning mellan trådarna

Min tanke med arbetsfördelningen är att alla trådar kör samma kod(samma funktioner) och stannar vid delade resurser och väntar (t.ex. kön) med ett ’’mutex_lock’’. Sedan för att inte dom ska köra på och försöka ta av kön när den är tom så har jag både en if sats som förhindrar att detta ska vara möjligt i kombination med en semanphor som kontrollerar så att ingen tråd kör på och skapar massor med konstiga minnesfel. Eftersom dom arbetar med samma sak allihopa så blir det lätt för dom att dela på arbetet desto fler dom är. Min ursprungliga tanke var att ha olika trådar som gör olika uppdelade arbetsuppgifter, jag tyckte dock i slutändan att detta enbart vart besvärligt när det skulle vara möjligt att köra koden för ett godtyckligt antal trådar.

4. Testkörningar

Jag gjorde ett bash script för att köra tester på koden som jag sen plottade 25 körning av gentemot tid i matlab. Som grund för datan så körde jag 10 testkörningar med 1 till 100 trådar. Programmet verkar fungera utmärkt för detta och man ser då detta plottade sammanhang:



Tydligt blev det att upp till 25 trådar så verkar alltid programmet bli snabbare och snabbare ju mer trådar vi använder. Med valgrind så blir det dock tydligt att det tar mer och mer minne

desto fler trådar vi kör, dock så får vi saker gjort snabbt. Så som jag förstått det så verkar detta sammanhang ha mycket att göra med maskinen man kör på. Desto mer processorkraft och ramminne desto snabbare går det med fler trådar att få ned söktiden ännu mer. Alla sammanlagda tester som är gjorda på itchy återfinnes som test1.txt till test10.txt.

5. Referenser

1. http://man7.org/linux/man-pages/man2/semop.2.html 2019-10-25
2. https://linux.die.net/man/3/pthread_mutex_lock 2019-10-25
3. https://www.cambro.umu.se/access/content/group/57250HT19-1/Laboration%204/mfind-specification.pdf 2019-10-25
4. http://man7.org/linux/man-pages/man2/semctl.2.html 2019-10-25
5. https://se.mathworks.com/help/matlab/ 2019-10-25
6. https://ss64.com/bash/time.html 2019-10-25
7. https://www.cambro.umu.se/portal/site/57250HT19-1 2019-10-25
8. Janlert, Lars-Erik., Wiberg, Torbjörn. 2000. *Datatyper och Algoritmer*. (2:a upplagan). Lund: Studentlitteratur AB.