OPERATIVSYSTEM 2014, ÖVNING 2, 13.11.2014

OBS. Rapport över utförda övningarinlämnas elektronisk på adressen https://abacus.abo.fi/ro.nsf.

1. Med kommandot `strace` (finns i de många linux-distributioner) kan man lista alla de anrop som görs till kärnan. Kompilera och exekvera HelloWorld-exemplet (nedan). Räkna upp alla systemanrop som görs, samt frekvensen av dessa. (2p)

```
// File helloworld.c
#include <stdio.h>

int main() {
  printf("Hello World\n");
}
// ------------------

Kompilering och körning
% gcc helloworld.c -o helloworld
% ./helloworld
% strace ./helloworld
```

2. Undersök effekten av kritiska sektioner i programmering. Använd koden HelloWorld (refererad nedan) som startpunkt och lägg till en ”kritisk sektion” i trådfunktionen (i detta fall PrintHello):

```
for (i=0; i<UPDATE_COUNT; i++) {
   tmp = count;
   count = tmp+1;
}
```

där **`count`** är en global variabel (denna måste du också lägga till). Om allt skulle gå rätt, skulle då när trådarna kört färdigt alltid ha count = UPDATE_COUNT*THREAD_COUNT, men i praktien är det ofta mindre. Redovisa resultaten i en tabell med kolumner UPDATE_COUNT, THREAD_COUNT, MISSED UPDATES (%). Finns det någon systematik?

PS. kom ihåg att vänta på att trådarna är färdiga (antingen via `sleep()`, eller via `pthread_join()`).

Ledning: använd Posix threads, tutorial finns på
https://computing.llnl.gov/tutorials/pthreads/
och HelloWorld för pthreads (som exempel/start) på
https://computing.llnl.gov/tutorials/pthreads/samples/hello.c

(6p)

3. Ett soft-realtidssystem har fyra periodiska händelser med perioder om 50, 100, *y* och 250 ms, där *y* är 10 * dag i månaden för ditt födelsedatum (t.ex *y*=170 för födelsedag 17.7). Anta att de fyra händelserna kräver 15, 20, 10 och *x* ms processortid respektive. Vilket är det största värde på *x* för vilket systemet är skedulerbart? (4p)

    a. Beräkna och motivera. (2p)
    b. Simulera systemed med hjälp av **`pthreads`** (jfr uppgift 2). Skapa en tråd för varje periodisk händelse. Trådarna kör en oändlig loop där
        1) Processortid simuleras genom att köra en tom loop som väntar tills en viss mängds processortid använts (funktionen **`clock()`** ger hur mycket tid som använts)
        2) Väntetid (dvs. period – processortid) skapas genom **`usleep()`**

    Kör systemet. Observera med hjälp av linux / unix top hur mycket processortid simuleringen tar. Om det är mer än 100% (om man använder en flerkärneprocessor) betyder det att systemet inte är skedulerbart på en normal enkärneprocessor. (6p)