ESWE

Embedded Software Engineering

bd.thrijswijk.nl

THRijswijk

ESWE

overzicht

- Kwartaal 1
  - C1 realtime embedded programming
    - 2 CP, tentamen telt voor 35% mee
    - Bd = Harry Broeders
  - C2 distributed programming
    - 1½ CP, tentamen telt voor 30% mee
    - Snb = Bart Snijder
  - P1 practicum distributed and realtime programming
    - 1½ CP, moet voldoende zijn, Bd
- Kwartaal 2
  - C3 software engineering
    - 2 CP, tentamen telt voor 35% mee.
    - Scp = Paul Schepens
  - P2 practicum software engineering
    - 1 CP, moet voldoende zijn, Scp

THRijswijk

Werkvormen

C1 + P1 = **98** SBU

- 14 uur ESWE1C1 theorie
- 14 uur ESWE1P1 practicum
- 70 uur zelfstudie

Inhoud C1

- concurrent programming (threads en scheduling)
- real-time OS (POSIX, QNX) and real-time language (RTJava)
- synchronisation and communication
- reliability and fault tolerance

THRijswijk

Leermiddelen

- **Boek**
  - Real-Time Systems and Programming Languages (Third Edition), Alan Burns and Andy Wellings, ISBN: 0201729881
  - QNX Neutino 2, Robert Krten (kun je bij mij lenen)
- **http://bd.thrijswijk.nl/eswe1/**
  - uitgebreide **studiewijzer**
  - sheets
  - **practicumhandleiding**
  - practicumdocumentatie
  - sourcecode van alle voorbeelden
  - links

THRijswijk

Real-time systeem

definitie

- Systeem waarvan de **reactietijd** op een inputverandering **voorspelbaar** is.
- ... zie boek

THRijswijk

Real-time systeem

Resultaat (response) moet niet alleen **correct** zijn maar ook op **tijd**!

- **hard** real-time
  - missen van een deadline is fataal
- **soft** real-time
  - missen van een deadline is ongewenst
- interactief
  - er zijn geen expliciete deadlines maar wachten is wel irritant

THRijswijk

Real-time systeem

voorbeelden

- Procesbesturing (meet en regeltechniek)
- Productie besturingssysteem (industriële automatisering)
- **Embedded systemen**
  - ABS (Anti-Blokeer-Systeem)
  - pacemaker
  - besturing kruisraket
  - kopieer apparaat
  - DVD recorder

THRijswijk

Real-time systeem

karakteristieken

- Groot en complex (niet altijd)
  - onderhoudbaar: uitbreidbaar, aanpasbaar en herbruikbaar
- Betrouwbaar en veilig
  - intensive care apparatuur
  - kerncentrale
  - automatische piloot
- Concurrent gedrag
  - multitasking, multiprocessor, distributed
  - RTOS of RTL moet dit ondersteunen
- Real-time faciliteiten
  - taak op bepaalde tijd starten, taak binnen bepaalde tijd afronden
  - RTOS of RTL moet dit ondersteunen
- Interactie met hardware
- ...

THRijswijk

Real-time systeem

life cycle

- Idee
- Analyse
  - kwartaal 2: ESWEC2 en P2
- Ontwerp
  - kwartaal 2: ESWEC2 en P2
- **Implementatie** boek 2.5
  - assembly (liever niet)
  - sequentiele programmeertalen: C en C++ (concurrency via RTOS)
  - concurrent programmeertalen: Java en RTJ
- **Testen** boek 2.6
  - concurrency is erg moeilijk te testen
  - simulatoren (om veiligheid en betrouwbaarheid te testen) kerncentrale, space shuttle
- Onderhoud

THRijswijk

Concurrent programming

potentieel parallelisme

- single processor system (multitasking)
- multi processor system met gedeeld geheugen (SMP)
- distributed system (ESWE1C2)

THRijswijk

10

Concurrent programming

waarom?

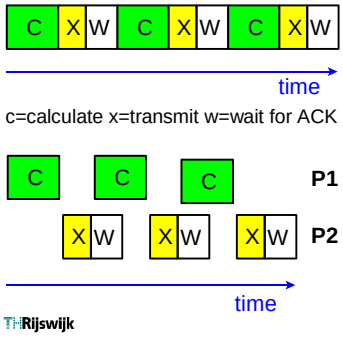
- Benutten van parallelisme in multiprocessor of distributed systeem.
  - zoek je weg in een doolhof
  - vergelijken van vingerafdrukken
- Programma model komt overeen met de werkelijkheid.
- Processor beter benutten op single processor systeem.

THRijswijk

11

Concurrent programming

processor beter benutten op single processor systeem.



THRijswijk

12

Threads

light weight process

- **process**
  - eigen stack en eigen data
    - = veilig
  - eigen memory map
    - eigen virtueel geheugen = veilig
    - communicatie = traag
  - process switch is traag (**zwaar**)
    - cache flush, MMU TLB flush
- **thread**
  - eigen stack en gedeeld geheugen met andere threads in hetzelfde process
    - = snel
  - gedeelde memory map binnen hetzelfde process
    - communicatie = snel
  - thread switch is snel (**licht**) binnen hetzelfde process
    - geen flushes

THRijswijk

13

Concurrent programming

implementaties

- Sequentiële programmeertaal (C of C++) **+** OS (Linux, Win32)
  - portable als taal en OS portable zijn IEEE **POSIX 1003**
  - meerdere talen combineren in 1 programma is mogelijk
- Concurrent programmeertaal (ADA, **Java** of C#)
  - beter leesbaar
  - beter onderhoudbaar
  - portable als taal portable is
- Middleware (RPC, **RMI**, CORBA) (ESWE1C2)

THRijswijk

14

Concurrent programming

fundamentele problemen

- Hoe kun je processen / threads beheren?
  - support in OS
  - support in programmeertaal
- Hoe kunnen processen / threads communiceren
  - IPC = Inter Process Communication (term wordt ook voor communicatie tussen threads gebruikt)
- Hoe kunnen processen / threads synchroniseren?
  - IPC zonder dataoverdracht.

THRijswijk

15

Concurrent programming

aspecten

- Structuur
  - statisch / dynamisch
- Niveau
  - genest / plat
- Granularity (korreligheid)
  - grof / fijn
- Initialisatie
  - parent child relatie
  - ouder brengt kind tot leven
- Termination
  - guardian dependant relatie
  - beschermer (voogd) kan pas sterven als afhankelijke (toevertrouwde) overleden is

THRijswijk

16

Concurrent OOP

object oriënted concurrency

- **actieve** objecten
  - eigen thread of process
  - versturen zelf actief (**spontaan**) messages
- **passieve** objecten
  - reageren op binnenkomende messages
  - kunnen als **reactie**:
    - zelf message versturen
    - toestand van thread of process veranderen (b.v. van running naar waiting).
- B.v. object die toegang tot resource beveiligd
  - actief: server
  - passief: protected resource

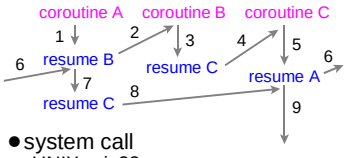
THRijswijk

17

Concurrent programming

representatie

- coroutine
  - coöperatief (eerste Apple, win16)



- system call
  - UNIX, win32
- concurrent blok
  - concurrent Pascal, Occam
  - cobegin s1; s2; s3 coend
- expliciete declaratie
  - ADA, Java

THRijswijk

18

Concurrent execution

IEEE POSIX 1003.1-2001

- fork() en wait()
- posix_spawn
  - combinatie van fork(), exec() en wait()
- pthreads

- Documentatie:
  - IEEE Std 1003.1-2001 = The Open Group Base Specifications Issue 6
    http://www.unix.org/single_unix_specification/
  - QNX documentation:
    http://www.qnx.com/developer/docs/

THRijswijk

fork

Hoeveel processes?

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
   pid_t pids[3];
   int i;
   for (i=0; i<3; ++i) {
      fflush(stdout); // waarom?
      pids[i]=fork();
      if (pids[i]==-1) {
         perror("fork");
         exit(EXIT_FAILURE);
      }
      if (pids[i]==0) { // child
         printf("child pid=%d\n", getpid());
      }
      else { // parent
         printf("parent pid=%d\n", getpid());
         waitpid(pids[i], NULL, 0);
      }
   }
   return EXIT_SUCCESS;
}
```

THRijswijk