TDIU16

Föreläsning 0

Filip Strömbäck, Klas Arvidsson

LINKÖPING UNIVERSITY

LINKÖPING UNIVERSITY

Resurser

- Kurshemsida: `http://www.ida.liu.se/~TDIU16/`
- Litteratur: *Operating System Concepts* av Silbertschatz, Galvin, Gagne och *Introduction to Concurrent Programming* (online)

Examinator	Filip Strömbäck	
Kursassistent	Dag Jönsson	
Assistenter	Love Arreborn	Eric Ekström
	Malte Hammar	Malte Nilsson
	Alex Nordgren	Jacob Törnberg Zeinetz
Administratör	Emilie Lind	

Mål med kursen

Tillämpa teori från TDIU11:

- Hur ser en "vanlig" dator ut?
- Hur körs program?
- Vilka viktiga resurser finns för programkörning?
- Varför finns operativsystem?
- Vad gör operativsystemet? När körs det?

Mål med kursen

Nytt innehåll:

- Multitrådning
- Synkronisering
- Programmering i C

Praktisk erfarenhet:

- Programmera i större system
- Läsa, förstå och modifiera kod som andra har skrivit
- Felsökning

FÖ, SE, LA

- Föreläsningar
  - Slides är inte avsedda att gå att läsa självständigt
  - Ställ frågor
  - Svara (gärna fel) på ställda frågor
- Seminarier
  - Bidra med lösningsidéer till labb 3 och 4
  - Ställ ännu mer frågor
- Lab
  - Arbeta så långt som möjligt på egen hand
  - Ställ frågor/demonstrera på labpassen
  - Långa/komplexa frågor på passen – ofta hög belastning

Examination

LAB1 Laborationer, 3hp (U, G)

- Ett ofärdigt operativsystem (Pintos) skrivet i C
- Redovisas muntligt under labpass
- Mjuka deadlines ger bonus
- Två hårda deadlines: 1 i mitten av kursen, 1 i slutet av kursen
- Eget arbete utöver schemalagd tid krävs

DAT1 Skriftlig tentamen med dator, 1hp (U, 3, 4, 5)

- LAB1 (speciellt uppg. 4) är förberedande för tentan

Labanmälan

- Arbete sker parvis som vanligt.
- Två grupper efter program:
  1. DI2, EL2
  2. IP2
- Finns också grupper för de som letar efter någon att jobba med.
- Labanmälan öppen tom. 7/4, anmäl er snarast!

Planering

Vecka	Fö/Se	Lab	Deadline
14	Fö: C + Syscall (+påsk)	1ab: C[1], 1c: intro	–
15	–	2: Systemanrop	–
16	Fö + Se: Semaforen	3ab: Processhantering	1, 2
17	Fö: Lås, cond, deadlock	3cd: Processhantering	–
18	Se: Synkronisering	4ab: Synkronisering	3
19	Fö: Låsimplementation	4bc: Synkronisering	–
20	–	4: Synkronisering, 5: Säkerhet	–
21	–	5: Säkerhet	4, 5

[1] lämpligt att demonstrera första passet

Tid i kursen

- Kursen är 4hp $\approx$ 100h ($=$ 2,5 *hela* arbetsveckor)
- Du kommer att behöva alla 100h. Planera din tid så att du arbetar med kursen varje vecka (ex.vis enligt planeringen).
- Det finns fristående uppgifter ni kan börja med ifall ni kör fast.

Värt att tänka på

- Du skriver kod i *kernelmode*: du får göra precis vad du vill.
- Du får göra "fel" som skulle krasha usermode. Finns ingen kernel som hindrar dig!
- Alltså: Fundera först, skriv kod sen. Det spar *mycket* felsökningstid!

Kommentarer från tidigare år

- Givande labbserie, men tar tid (speciellt om felsökning krävs)
- Mycket kod att sätta sig in i, svårt att veta vad som är ”rätt”
- Bra med “standalone”-delar innan Pintos

Förändringar:

- Seminarie 2 tidigarelagt (blir “intro till synkronisering”, inte “inför tentan”)
- Nytt kursmaterial om synkronisering (komplement till OS-boken)
- Möjlighet att köra en delmängd av automatiska tester enklare
- Uppdaterad version av kodskelettet

LINKÖPING UNIVERSITY

C och C++ – vad är skillnaden?

I allmänhet är C ett mycket tunnare lager ovanpå hårdvaran än C++ (byggt för att skriva UNIX)

Det mesta i C är giltigt i C++, men det finns små skillnader.

Notera att `struct` och `union` har separata namnrymder i C. Går att komma runt med `typedef`.

Notera: `int foo();` $\not\Longleftrightarrow$ `int foo(void);`

C och C++ – vad är skillnaden?

C++	C
`cout`	`printf`
klasser	`struct`
referenser	pekare
`vector`, `string`	arrayer + biblioteksfunktioner
`private`	definiera i c-fil, `static`
statisk datamedlem	`extern`
`new`, `delete`	`malloc`, `free`
virtuella funktioner	funktionspekare
klassmallar	`void *`, makron (se Pintos lista)
exceptions	returvärden

Utmatning

```
int main() {
    int a = 1;
    unsigned int b = 2;
    float c = 3.5f;
    double d = 5.8;
    char e = 'A';
    int *f = &a;
    const char *g = "hello";

    cout << a << b << c << d
         << e << f << g << endl;
}
```

Utmatning

```
int main() {
    int a = 1;
    unsigned int b = 2;
    float c = 3.5f;
    double d = 5.8;
    char e = 'A';
    int *f = &a;
    const char *g = "hello";

    printf("%d %u %f %f %c %p %s\n",
           a, b, c, d, e, f, g);
}
```

Utmatning

```
int main() {
    int a = 10;
    const char *b = "hello";

    printf("%6d\n%06d\n%6s\n", a, a, b);
    // Ger:
    //     10
    // 000010
    //  hello
}
```

Exempel på pekararitmetik

I visualiseringsverktyget: *File* → *Open example...* titta sedan i katalogen `pointers`:

- `locals.c`
- `struct.c`
- `array.c`
- `array_malloc.c`
- `struct_array.c`

Första veckorna

Innan första passet:

- Installera Pintos
- GDB-labbar
- (Associativ container)

Första veckan:

- Mål: Klar med labb 1

Notera: Håll implementationen av 1c (associativ container) så enkel som möjligt. Det underlättar senare!

Vecka 2:

- Labb 2: Systemanrop
  - a: Uppvärmning: `halt`, `exit`
  - b: Input-output: `read`, `write`
  - c: Filhantering: `open`, `close`, `create`, ...
- Notera: Funktionaliteten finns i Pintos. Målet är att göra det möjligt att använda funktionalitet som redan finns från usermode!

Filip Strömbäck, Klas Arvidsson

www.liu.se

LINKÖPING UNIVERSITY