UNIVERSITETET I OSLO
DET MATEMATISK-NATURVITENSKAPELIGE FAKULTET

Programmeringsspråket Perl (del I)

- Bakgrunn
  - Hvem laget Perl, og hvorfor?
- Oversikt over Perl
  - Setninger, uttrykk, datatyper, datastrukturer, ...
- Informasjon om Perl

INF3110/4110

Bakgrunnen

Den opprinnelige Perl ble laget av *Larry Wall* i 1986. Han jobbet som systemprogrammerer og manglet verktøy til de oppgavene han hadde.

Språket het først *Pearl* («Practical extraction and report language») men måtte bytte navn fordi det fantes et grafisk språk med samme navn.

Larry Wall har (i følge ham selv) følgende egenskaper:

- **lat**

  Dette har gjort Perl ordknapt og kompakt.

- **utålmodig**

  Perl er raskt.

- **overmodig**

  Han har laget et språk som skal kunne brukes til omtrent alt.

Hva brukes Perl til?

Perl brukes mest til følgende oppgaver:

Tekstbehandling pga meget kraftige operasjoner for dette.

Systemprogrammering pga god aksess til mekanismer i operativsystemet, spesielt Unix. Programmer som preview og setbb er skrevet i Perl.

VVV-programmering pga de to foregående punktene. Dessuten er Perl meget portabel; man trenger ikke engang rekompilere.

Én-linjes programmer som

```
> perl -ne 'print $_ if /(\S+)\ssetcmyk/ && $1>0.5' intro.ps
setcmykcolor}DC/Brown{0 0.81 1 0.60 setcmykcolor}DC
    0 0 0 1 setcmykcolor systemdict /currentgray
```

Mitt møte med Perl

Etter å ha skrevet og spesielt vedlikeholdt print i noen år, var jeg meget frustrert over Unix´ standard skriptspråk *Bourne-shell* (sh):

- Det var tildels store forskjeller på sh og hjelpeprogrammene på de ulike Unix-er.
- Det var umulig å håndtere filnavn som inneholdt #, blanke samt æ, ø og å på en skikkelig måte.
- sh er ikke voldsomt raskt.
- Syntaksen† er eiendommelig og ofte kronglete å bruke.

På alle punkter (unntatt muligens det siste) er Perl en klar forbedring.

† Språkets grammatikk, altså reglene for hvorledes for eksempel setninger skal se ut.

En kort oversikt over Perl

Vi lar Perl presentere seg selv:

```
#! /store/bin/perl -w

print "Hallo, alle sammen!\n";
```

Perl er interpretert

I motsetning til de fleste språk er Perl interpretert: tolkeren /store/bin/perl leser kildekoden, oversetter den til et internt format, og utfører den.

Ved å sette riktig beskyttelse, kan man bruke filnavnet som kommando:

```
> chmod a+x hallo
> hallo
Hallo, alle sammen!
```

Kommentarer

Kommentarer i Perl angis med en # og varer til slutten av linjen.

Tekster

Tekster angis vanligvis med "...". De kan inneholde «\n» for å angi linjeskift.

Datatyper

Perl har bare to datatyper:

tall lagret som dobbeltpresisjons flyttall (double i C eller Java). Dette fungerer også helt fint for alle vanlige heltall; brukeren trenger bare tenke på verdiene som *tall*.

tekst av vilkårlig lengde som kan inneholde alle tegn.

Det er automatisk konvertering mellom disse to:

```
if ("124" < 1000) ...
```

Perl mangler noen vanlige datatyper:

- For logiske verdier (boolean i Java) benyttes et tall eller en tekst:
  - Verdiene 0, "0" og "" regnes som **usann**,
  - alle andre verdier regnes som **sann**.
- For enkelttegn (char i C og Java) brukes en tekst av lengde 1.

Enkle variable

Enkle («scalar») variable har alltid en $ først i navnet. Det er unødvendig å deklarere dem:

```
$antall = 4;
$hilsen = "Hei";

print $hilsen, ", alle ", $antall, "!\n";
```

Siden operatoren . skjøter to tekster (som + i Java eller strcat i C), kan vi også skrive

```
print $hilsen . ", alle " . $antall . "!\n";
```

($antall blir da automatisk konvertert til en tekst.)

Variable kan også forekomme i tekster:

```
print "$hilsen, alle $antall!\n";
```

Denne overfloden av muligheter er typisk for Perl.

Løsning: Velg den versjonen du selv synes er mest intuitiv (og gjerne også kortest).

Uttrykk

De vanlige operatorene fra C og Java finnes også i Perl:

- Aritmetiske operatorer: +, -, *, / og % (for resten ved en divisjon).
- Sammenligning av tall: ==, !=, <, <=, > og >=.
- Sammenligning av tekster: eq, ne, lt, le, gt og ge.

Perl har dessuten Cs operatorer for ± 1:

```
++$a og $a++  øker $a med 1
--$b og $b--  senker $b med 1
```

Følgende operatorer er til dels nye:

- Skjøting av tekster:

$$\text{"abc" . "x"} \overset{\text{gir}}{\Longrightarrow} \text{"abcx"}$$

- Gjentagelse av tekst:

$$\text{"abc" x 3} \overset{\text{gir}}{\Longrightarrow} \text{"abcabcabc"}$$

Setninger

De viktigste setningene i Perl er disse:

- die "melding\n"; vil avbryte kjøringen med en feilmelding.
- exit *n*; avslutter programmet med angitt statusverdi.
- print . . . ; benyttes ved utskrift.
- Alle uttrykk (spesielt tilordning som $x = 4;) er lovlige setninger.
- if-tester er som i Java og C. Det finnes også en unless-test med motsatt resultat:

```
if (x < 0) { . . . }
unless (x >= 0) { . . . }
```

- while-løkker er som i Java og C. Det finnes en «motsatt» variant: until.
- foreach-løkker går gjennom elementene i en liste (omtales senere):

```
foreach $x (@navn) {
  print $x;  ++$n;
  if ($n%5 == 0) { print "\n"; }
}
```

INF3110/4110

Setninger med betingelse

Alle vanlige setninger kan gis en betingelse (med if eller unless) som angir om setningen skal utføres eller ikke:

```
$x = -$x unless $x >= 0;
```

Denne varianten brukes der det føles naturlig.

Den implisitte variabelen $_

Variabelen $_ er meget spesiell:

> *Hvis vi utelater en variabel der det forventes en, mener vi $_.*

Eksempel

Disse to setningene betyr det samme:

```
print;
print $_;
```

Eksempel

Med disse hjelpemidlene kan løkken på ark 9 skrives noe mer kompakt:

```
foreach (@navn) {
  print;
  print "\n" unless (++$n%5);
}
```

Denne «videreutvilkingen» av et program er typisk for Perl.

Informasjon om Perl

- Boken *Learning Perl* av Randal Schwartz; O'Reilly & associates.
- Boken *Programming Perl* av Larry Wall, Tom Christiansen & Randal Schwartz; O'Reilly & associates.

  Dette er «kamelboken», selve standardverket om Perl. Den er uunnværlig for alle som skal bruke Perl seriøst.
- Filen /local/doc/perl/perl-ref.ps inneholder en meget nyttig 12-siders oversikt over Perl.
- Nyhetsgruppen no.it.programming.perl er bemannet med mange hjelpsomme lesere.