

KARLSTADS UNIVERSITET  
INSTITUTIONEN FÖR MATEMATIK OCH DATAVETENSKAP

TEORETISK DATALOGI  
DVGA17

---

# Automater och Språk

---

*Skriven av:*

Alexander FLOREAN  
florean.alexander@gmail.com  
Emanuel SVENSSON  
emansven100@student.kau.se

*Handledare:*

Kerstin ANDERSSON

28 november 2019



## Innehåll

<b>1</b>	<b>Översikt</b>	<b>2</b>
<b>2</b>	<b>Antaganden</b>	<b>2</b>
<b>3</b>	<b>Resultat</b>	<b>2</b>
3.1	bash . . . . .	2
3.2	grep . . . . .	3
3.3	expr . . . . .	4
<b>4</b>	<b>Lexikalanalys</b>	<b>5</b>
<b>5</b>	<b>Sammanfattning</b>	<b>6</b>

# 1 Översikt

Laborationen gick ut på att utforska unix-verktygen bash, grep och expr och hur dem använder sig av reguljära uttryck, samt att skapa en förenklad C-skanner med lexikalanayls.

## 2 Antaganden

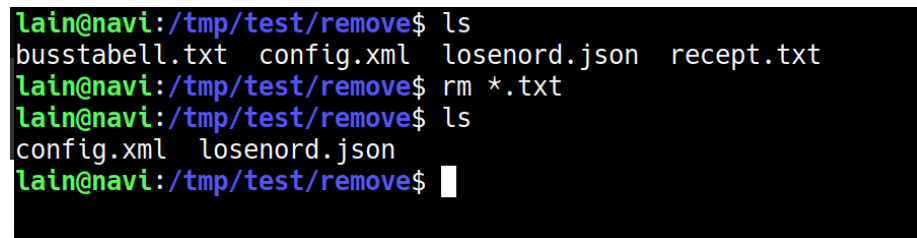
## 3 Resultat

### 3.1 bash

*bash* (Bourne-again shell) är en kommandotolk som är vanligt på unixlikenande system som Ubuntu och OSX.

**Funktionalitet** En stark funktionalitet hos Bash är dess kraftfulla förmåga att använda shellglobar för att matcha filnamn. Karaktären `*` används för att matcha alla karaktärer i en sträng.

**Exempel: matchning av filtyper**

A terminal window with a black background and green text. The prompt is 'lain@navi:/tmp/test/remove\$'. The first command is 'ls', which lists 'busstabell.txt', 'config.xml', 'losenord.json', and 'recept.txt'. The second command is 'rm \*.txt', which removes the two .txt files. The third command is 'ls', which now only lists 'config.xml' and 'losenord.json'.

```
lain@navi:/tmp/test/remove$ ls
busstabell.txt  config.xml  losenord.json  recept.txt
lain@navi:/tmp/test/remove$ rm *.txt
lain@navi:/tmp/test/remove$ ls
config.xml  losenord.json
lain@navi:/tmp/test/remove$
```

Figur 1: Filer raderas efter filtyper

I exemplet används `*` för att matcha alla filnamn som slutar med `.txt`, effektivt raderar kommandot alla textfiler.

**Exempel: matchning av alla filer**

```
lain@navi:/tmp/test$ ls
move  remove  remove_cont
lain@navi:/tmp/test$ ls remove/
config.xml  losenord.json
lain@navi:/tmp/test$ mv remove/* move/
lain@navi:/tmp/test$ cd move/
lain@navi:/tmp/test/move$ ls
config.xml  losenord.json
lain@navi:/tmp/test/move$
```

Figur 2: Filer flyttas från en katalog till en annan

Kommandot i figuren matchar alla filer i en katalog och flyttar över dem till en annan.

#### Exempel: matchning av innehåll i filnamn

```
lain@navi:/tmp/test/remove_cont$ ls
auto_sent_mail.kmn  config.xml  losenord.json  skapad_auto_config.txt  verification_auto.js
lain@navi:/tmp/test/remove_cont$ rm *auto*
lain@navi:/tmp/test/remove_cont$ ls
config.xml  losenord.json
lain@navi:/tmp/test/remove_cont$
```

Figur 3: Filer raderas innehåll i deras filnamn

I figuren visas hur man i bash kan använda globbing för att matcha filer som innehåller ett visst nyckelord och ta bort dem.

## 3.2 grep

*grep* är ett kommandotolkverktyg som används för att hitta textlinjer som matchar reguljära uttryck.

**Funktionalitet** *grep* tar emot ett reguljärt uttryck och söker sedan efter en linje som matchar och returnerar linjer med matchande delar upplyst. *grep* kan konfigureras till att söka i olika medier som i en fil, input från användaren som unix pipes. Den kan även söka igenom kataloger efter filer vars namn matchar ett reguljärt uttryck och sedan kolla ifall den finner en matchning i dess innehåll.

#### Exempel: sökning i fil

```
lain@navi:~/Nextcloud/univ/datastrukturer_och_algoritmer/labbar/Labb1/Lab1-perf$ grep -e 'int \*[b-k]' analyze.c
static int *create_sorted_list(int n){
static int *create_reverse_list(int n){
static int *create_random_list(int n){
static void create_quickbest_switch(const int *src, int *dest, int start, int stop, int *pos){
static int *create_quickbest_list(int n){
lain@navi:~/Nextcloud/univ/datastrukturer_och_algoritmer/labbar/Labb1/Lab1-perf$
```

Figur 4: Sökning i fil

Sökning i en c-fil efter linjer som innehåller deklarationer med int-pekare vars namn börjare med bokstäver i alfabetet från b till k.

Exempel: sökning i input

```
lain@navi:~$ ps -e | grep mullvad
1407 ?          00:00:07 mullvad-daemon
1757 ?          00:00:07 mullvad-gui
1809 ?          00:00:00 mullvad-gui
1878 ?          00:00:01 mullvad-gui
1889 ?          00:00:23 mullvad-gui
lain@navi:~$ sudo kill -9 1407
```

Figur 5: Sökning i input från annat program

I exemplet tar *grep* emot input från ett annat program för att hitta PID:en till en process med ett namn innehållandes mullvad. Kommandot *ps -e* har som output alla processer som körs på systemet som hade varit för mycket att söka igenom för hand.

Exempel: sökning efter fil med uttryck

```
lain@navi:~$ grep -e [a-c].random -r .
./.thunderbird/ow7yadqn.default/ImapMail/imap.gmail-1.com/INBOX: <
td align="left" valign="top" style="word-break:keep-all;-ms-word-break:keep-all;font
-size:12px; line-height:19px; color:#94724D; padding:0px 0px 10px 12px; font-family:
Verdana, Geneva, sans-serif;" class="mobileShow">For the next 4 days, you can treat
yourself to a Mystery Skin. Each Mystery Skin will deliver a randomly selected un-o
wned skin currently in the store at 520 RP or higher.</td>
./.thunderbird/ow7yadqn.default/ImapMail/imap.gmail-1.com/INBOX:X-MaxCode-Template:
email-cc-random-deposit-notice
^C
lain@navi:~$
```

Figur 6: Sökning efter filer som har linjer som matchar uttryck

*grep* kan söka igenom en katalog efter filer vars innehåll matchar ett uttryck

### 3.3 expr

**Funktionalitet** *expr* är ett kommandotolkverktyg som utvärderar uttryck med tal, strängar och jämförelser. För att undvika att kommandotolken ut-

värderar uttrycken måste speciella karaktärer utflys med

#### Exempel: matematiskt uttryck

```
lain@navi:~$ expr \( \( 20 + 5 \) \% 23 \) \* 24
48
lain@navi:~$ █
```

Figur 7: Utvärdering av matematiskt uttryck

I figuren demonstreras hur aritmetik och uträkningar kan utföras.

#### Exempel: blandat uttryck

```
lain@navi:~$ expr \( index storbritannien brit \< 5 \) \* 10
10
lain@navi:~$ █
```

Figur 8: Utvärdering av blandat uttryck

I exemplet utvärderas uttryck med både matematiska tal och karaktärer. Uttrycket mäter först vart i strängen storbrittaniensom britförekommer och jämför sedan om indexet är mindre än 5. Resultatet blir sant, en 1:a, vilket sedan multipliceras med 10.

#### Exempel: reguljära uttryck

```
lain@navi:~$ expr abbbbbbbba : '[a-z]b\+'
9
lain@navi:~$ █
```

Figur 9: Utvärdering av reguljärt uttryck

I figuren visar ett reguljärt uttryck som räknar hur många gånger det förekommer bokstaven b en eller flera gånger med bokstäver från a-z framför.

## 4 Lexikalanalys

Eftersom det skulle skapas ett språk för att skanna *fak.c*, valdes att enbart det alfabet som *fak.c* använder. Detta gjordes genom att titta på *fak.c*. *pas.l* användes som en mall för att utveckla språket för specifikt *fak.c*

```

#include <stdio.h>
#include <stdlib.h>

int fak(int n) {
    if (n == 0)
        return 1;
    else
        return n * fak(n - 1);
}

int main(int argc, char *argv[]) {
    int n;
    if (argc != 2) {
        fprintf(stderr, "usage %s <n> \n\n", argv[0]);
        exit(1);
    }
    n = atoi(argv[1]);
    printf("n = %3d n! = %3d\n", n, fak(n));
    return 0;
}

```

Figur 10: fak.c koden

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

#define MAX_LEN 100
#define MAX_VARS 100

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "usage %s <file> \n\n", argv[0]);
        exit(1);
    }
    FILE *f = fopen(argv[1], "r");
    if (!f) {
        fprintf(stderr, "cannot open file %s\n", argv[1]);
        exit(1);
    }
    char line[MAX_LEN];
    while (fgets(line, MAX_LEN, f)) {
        // Tokenize the line
        char *token = strtok(line, " \t\n");
        while (token) {
            // Check if it's a keyword
            if (iskeyword(token)) {
                // ...
            }
            // Check if it's a variable
            else if (isalpha(token[0]) || token[0] == '_') {
                // ...
            }
            // Check if it's an operator
            else if (isop(token)) {
                // ...
            }
            // Check if it's a constant
            else if (isdigit(token[0]) || token[0] == '.' || token[0] == '-') {
                // ...
            }
            // Check if it's an unrecognized character
            else {
                // ...
            }
            token = strtok(NULL, " \t\n");
        }
    }
    fclose(f);
    return 0;
}

// iskeyword
int iskeyword(char *token) {
    // ...
}

// isop
int isop(char *token) {
    // ...
}

// isconst
int isconst(char *token) {
    // ...
}

// unrecognized
int unrecognized(char *token) {
    // ...
}

```

Figur 11: fak.l språket

*#include* I Figure[11] är *#include* ett token för importering av bibliotek.

*ID\*.h*

## 5 Sammanfattning

Sammanfatta ert projekt med hur det har gått och vad ni har lärt er. Beskriv alternativa lösningar samt fördelar och/eller nackdelar med dessa och varför ni valt den lösning ni har. Ta också med ungefärlig tidsåtgång för de olika momenten.