SYSTEMNÄRA PROGRAMMERING

STARTAR 13:15

5DV088 – HT21

EXAMINATION

- Tentamen (U/3/4/5)
  - Får ha med sig en C-bok
    - Se Canvas för godkända böcker
  - Tentamensregler
    - Se länk på Canvas

VAR?

- Tisdag 26/10 08:00 – 12.00
- Östra paviljongen
  - Skrivsal 2
    - Entré 3

NYA TILLFÄLLIGA RUTINER

- Studenten ska stanna hemma vid sjukdom
  - Om studenten är sjuk eller har förkylningssymptom ska hen stanna hemma. Om studenten kommer till tentamenssalen och visar förkylningssymptom blir hen hemskickad.
- Studenten behöver vara anmäld och ha legitimation
  - Endast studenter som har giltig legitimation och är anmälda (och därmed finns på anmälningslistan till tentamen) får skriva tentamen.
- Som jag har förstått det så gäller detta fortfarande

NYA TILLFÄLLIGA RUTINER

- **Det här gäller vid tentamen:**
  - Lärare får inte besöka salen under tentamens gång.
  - Studenten kommer inte att kunna ringa lärare om oklarheter i tentan. Vid oklarheter gör studenten en kommentar om det i tentan. *Oklart om detta gäller fortfarande!*

TENTAN

- **Hjälpmedel**: **EN** av följande böcker

  Bilting & Skansholm: "*Vägen till C*" ELLER
  J.R. Hanly & E.B. Koffman: "*Problem Solving and Program Design in C*" ELLER
  A. Kelley & I. Pohl: "*A Book on C*" ELLER
  Brian W. Kernighan & Dennis M. Ritchie: "*C Programming Language*" ELLER
  Brian W. Kernighan & Dennis M. Ritchie: "*Programmeringsspråket C*" ELLER
  Håkan Strömberg: "*C genom ett nyckelhål*"
- Maxpoäng: 40 (Gräns för 3: **20**p (50%), 4: **26**p (65%), 5: **32**p (80%))
- Börja varje uppgift på *nytt* blad och fyll i koden på *varje* blad.
- Inget besök i tentasalen.
  Vid tveksamheter – beskriv er tolkning.
- Tänk på att man kan få poäng även om man inte lyckas lösa hela uppgiften.

  Lycka Till!

Funktionsprototyper

```
int     execl(const char *pathname, const char *arg0, .../* (char *) 0 */);
                <unistd.h>
                Returns: -1 on error, no return on success
int     execle(const char *pathname, const char *arg0, .../* (char *) 0 , char *const envp[] */);
                <unistd.h>
                Returns: -1 on error, no return on success
int     execlp(const char *filename, const char *arg0, .../* (char *) 0 */);
                <unistd.h>
                Returns: -1 on error, no return on success
int     execv(const char *pathname, char *const argv[]);
                <unistd.h>
                Returns: -1 on error, no return on success
int     execve(const char *pathname, char *const argv[], char *const envp[]);
                <unistd.h>
                Returns: -1 on error, no return on success
int     execvp(const char *filename, char *const argv[]);
                <unistd.h>
                Returns: -1 on error, no return on success
void    exit(int status);
                <stdlib.h>
                This function never returns
int     fclose(FILE *fp);
                <stdio.h>
                Returns: 0 if OK, -1 on error
int     feof(FILE *fp);
                <stdio.h>
                Returns: nonzero (true) if end of file on stream, 0 (false) otherwise
int     fgetc(FILE *fp);
                <stdio.h>
                Returns: next character if OK, EOF on end of file or error
int     fgets(char *buf, int n, FILE *fp);
                <stdio.h>
                Returns: buf if OK, NULL on end of file or error
FILE    *fopen(const char *filename, const char *type);
                <stdio.h>
                type: "r", "w", "a", "r+", "w+", "a+"
                Returns: file pointer if OK, NULL on error
```

TENTAN

- Läs frågorna noga och svara på det som efterfrågas.
- Liten programmeringsuppgift
  - Jag "handkompilerar" koden, så ta med *allt*. (Dock behöver man inte ta med alla includefiler så länge man indikerar att det kan behövas fler.)
  - Testa saker och ting.
  - Eftersom funktionsprototyper finns ska de användas korrekt.
  - Behöver man någon annan funktion som man inte kommer ihåg vad den heter, beskriv vad gör och vad den har för argument.
  - Läs igenom extra noga så att du löser rätt problem.

VANLIGA PROBLEM

- Användande av exec-funktioner:

  Vill starta: `ls -l myfile mydir`

```
execlp("ls", "ls", "-l", "myfile", "mydir", (char *)0);
```

eller

```
char *arguments[5];
arguments[0] = "ls";
arguments[1] = "-l";
arguments[2] = "myfile";
arguments[3] = "mydir";
arguments[4] = NULL;
execvp("ls", arguments);
```

VANLIGA PROBLEM

- Allokering av minne

Uppgift 2 (4 p)

Givet följande kodavsnitt:

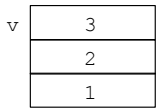
```
int v[] = {3, 2, 1};
char a[] = "xyz";
char *b = malloc(5);
char *c = b + *(v + 1);
b = "how?";
```

Rita en bild av datastrukturerna *a*, *b*, *c* och *v* (med innehåll), där man tydligt ser vad som allokerats och använd pilar för att markera hur pekarvärden är satta (efter kodsnutten).

VANLIGA PROBLEM

- Allokering av minne

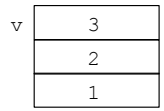
```
int v[] = {3, 2, 1};
char a[] = "xyz";
char *b = malloc(5);
char *c = b + *(v + 1);
b = "how?";
```

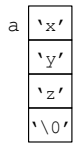


VANLIGA PROBLEM

- Allokering av minne

```
int v[] = {3, 2, 1};
char a[] = "xyz";
char *b = malloc(5);
char *c = b + *(v + 1);
b = "how?";
```

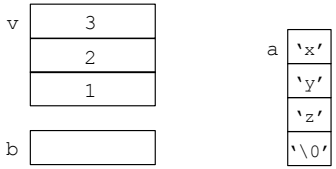




VANLIGA PROBLEM

- Allokering av minne

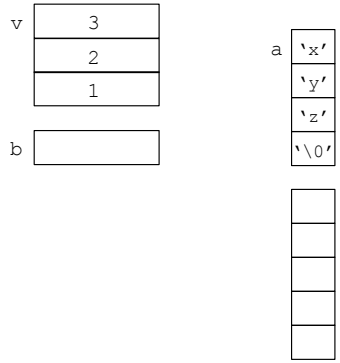
```
int v[] = {3, 2, 1};
char a[] = "xyz";
char *b = malloc(5);
char *c = b + *(v + 1);
b = "how?";
```



VANLIGA PROBLEM

- Allokering av minne

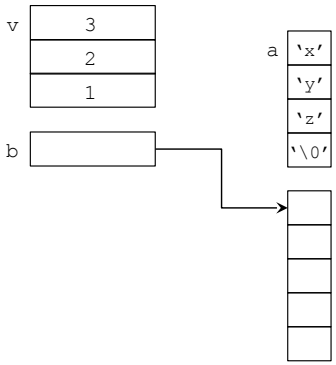
```
int v[] = {3, 2, 1};
char a[] = "xyz";
char *b = malloc(5);
char *c = b + *(v + 1);
b = "how?";
```

VANLIGA PROBLEM

- Allokering av minne

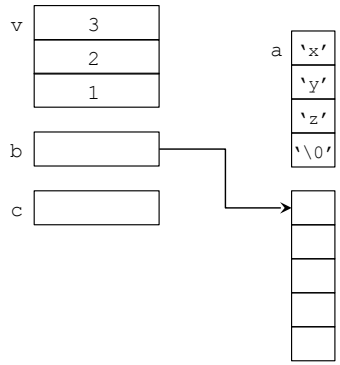
```
int v[] = {3, 2, 1};
char a[] = "xyz";
char *b = malloc(5);
char *c = b + *(v + 1);
b = "how?";
```

VANLIGA PROBLEM

- Allokering av minne

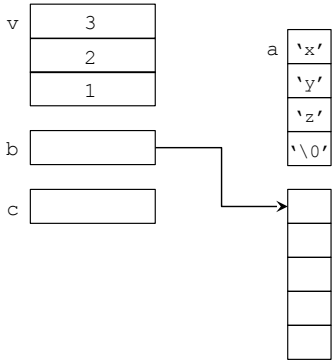
```
int v[] = {3, 2, 1};
char a[] = "xyz";
char *b = malloc(5);
char *c = b + *(v + 1);
b = "how?";
```

VANLIGA PROBLEM

- Allokering av minne

```
int v[] = {3, 2, 1};
char a[] = "xyz";
char *b = malloc(5);
char *c = b + *(v + 1);
b = "how?";
```

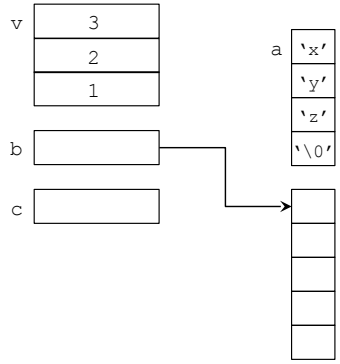


VANLIGA PROBLEM

- Allokering av minne

```
int v[] = {3, 2, 1};
char a[] = "xyz";
char *b = malloc(5);
char *c = b + *(v + 1);
b = "how?";
```

Blir `v[1]` som är `2`



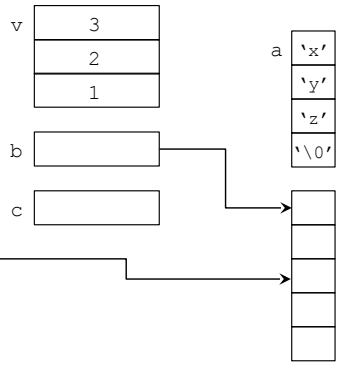
VANLIGA PROBLEM

- Allokering av minne

```
int v[] = {3, 2, 1};
char a[] = "xyz";
char *b = malloc(5);
char *c = b + *(v + 1);
b = "how?";
```

Blir `v[1]` som är `2`

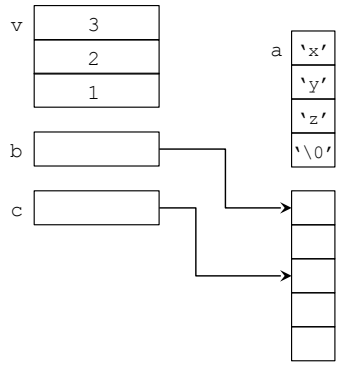
adressen `b + 2`



VANLIGA PROBLEM

- Allokering av minne

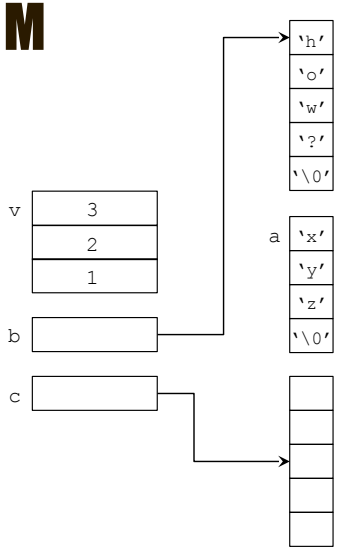
```
int v[] = {3, 2, 1};
char a[] = "xyz";
char *b = malloc(5);
char *c = b + *(v + 1);
b = "how?";
```



VANLIGA PROBLEM

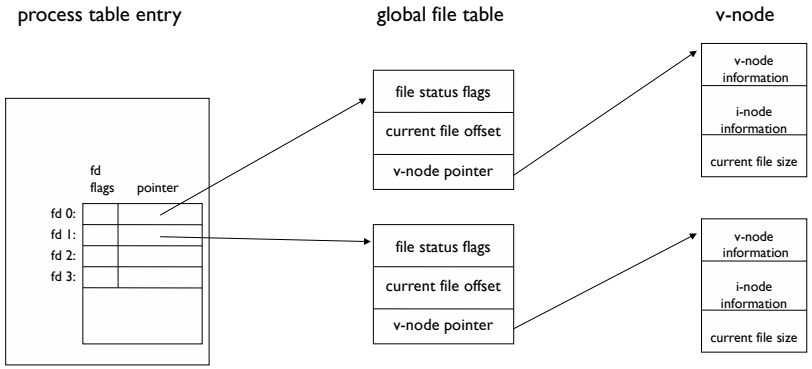
- Allokering av minne

```
int v[] = {3, 2, 1};
char a[] = "xyz";
char *b = malloc(5);
char *c = b + *(v + 1);
b = "how?";
```

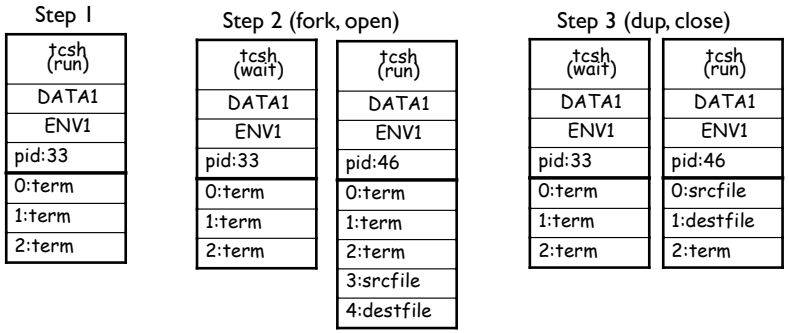


VANLIGA PROBLEM

- Vad skrivs ut?
  - Se add1.c
  - Se separat pdf



tr "[a-z]" "[A-Z]" < srcfile > destfile



tr ”[a-z]” ”[A-Z]” < srcfile > destfile

Step 4 (exec)

tcsh (wait)	tr (run)
DATA1	DATA2
ENV1	ENV1
pid:33	pid:46
0:term	0:srcfile
1:term	1:destfile
2:term	2:term

Step 5 (exit)

tcsh (run)
DATA1
ENV1
pid:33
0:term
1:term
2:term

who | grep jacob | wc -l

Se separat exempel

BIT-OPERATIONER

- Ganska frekvent använt I systemprogrammeringssammanhang
- Endast på heltalsoperander
- **&** Används för att “ta bort” (mask off) vissa bitar (1 om båda 1)
- | Används för att sätta vissa bitar (1 om någon är 1)
- **^** (1 om olika 0 i övrigt)
- Shiftoperatorerna n **<<** x och n **>>** x förflyttar bitarna i n x steg (x>0)
  - Left shift (**<<**) fyller på med nollor från vänster
  - Right shift (**>>**)
    - om unsigned alltid nollor
    - Signed 1 eller 0 beroende på maskin