C-Programmierung unter LINUX

Lernziele:
- Arbeitsverzeichnis einrichten (mkdir, cd)
- Quellcode editieren (Editor: joe)
- Quellode compilieren (gcc)
- Programme ausführen

Wiederholung:
- include-Dateien
- Arrays
- Datenstrukturen
- printf-Anweisung
- for-Schleife
- if-Abfrage

LINUX starten

LINUX starten → PC einschalten, Linux im Boot-Menü wählen

Anmelden → Nutzerkennung und Passwort sind auf dem PC notiert
Beispiel: Nutzer **`pc01`**, Passwort **`pc01`**

zur Orientierung: **`dir`** oder **`ls -l`** zeigt Dateien und Verzeichnisse
Beispiele:

```
drwx------   3 pc01 users   4096 Jan 1 1988  KDesktop
```

ist ein vom Administrator eingerichtetes Verzeichnis

```
drwxr-xr-x   3 pc01 users   4096 Jan 1 00:42 test
```

ist ein vom Nutzer eingerichtetes Verzeichnis

```
-rwxr-xr-x   3 pc01 users  13492 Jan 1 00:47 hello
```

ist ein vom Nutzer erzeugte ausführbare Datei

Arbeitsverzeichnis einrichten

Befehl: **`mkdir`** = make directory

Beispiel: **`mkdir meier`**

in das Verzeichnis wechseln: **`cd`** = change directory

Beispiel: **`cd meier`**
ergibt: **`pc01@pc01:~/meier>`**

zurück in das Hauptverzeichnis: **`cd ..`** (Leerzeichen beachten)

Verzeichnis löschen: **`rmdir`** = remake directory

Quellcode editieren

Editor: **`joe`**

wichtige Befehle:
- `Ctrl-k h` = Hilfe anzeigen
- `Ctrl-k d` = Datei sichern
- `Ctrl-k x` = Datei sichern und Editor verlassen
- `Ctrl-c` = Editor verlassen ohne zu sichern

Blöcke markieren und kopieren/verschieben:
- `Ctrl-k b` = Anfang des Blocks
- `Ctrl-k k` = Ende des Blocks
- `Ctrl-k m` = Block bewegen (move)
- `Ctrl-k c` = Block kopieren (copy)

ein kleines Beispielprogramm...

Eingabe mit Hilfe des Editors **joe**:

```
#include <stdio.h>

main()
  {
   printf("\n mein erstes C-Programm für LINUX\n");
  }
```

Speicherung in einer Datei mit der Dateierweiterung `c` :

```
Name of file to save: prog.c
```


Dateien kopieren

Kopierbefehl für das LINUX-Dateisystem: **`cp`** (copy)

Beispiel: `cp prog.c progo2.c`
→ kopiert die Datei `prog.c` in die Datei `prog2.c`

Datei auf eine DOS/Windows-Diskette kopieren: (m-Tools)

Beispiel: **`mcopy`** `prog.c a:`
→ kopiert die Datei `prog.c` auf eine Diskette im Laufwerk `a:`

weitere m-Tools: `mdir, mdel ...`
Infos dazu in den Manual-Pages: `man mdir ...`

Quellcode compilieren

Verwendung des GNU-C Compilers: **`gcc`**

Beispiel: **`gcc`** `-o prog prog.c`

Option `-o` gibt den Namen der ausführbaren Datei an
→ Die Datei `prog.c` wird übersetzt. Die ausführbare Datei ist `prog`

falls Fehler gefunden werden:
→ Editor `joe` erneut aufrufen, Programm verbessern ...

mit den Pfeiltasten ↑ könne die letzten Eingaben wiederholt werden !

Programme ausführen

Befehl: Name der ausführbaren Datei + Eingabetaste
Beispiel: `prog <Eingabe>`

Hinweise:
- die Datei muss ausführbar sein (Attribut `x`)
  Attribute ändern: `chmod +x prog` macht die Datei ausführbar
- die Datei muss nicht die Dateierweiterung `exe` oder `com` haben
- die Datei muss im aktuellen Verzeichnis oder Suchpfad sein
  Suchpfad anzeigen: `$PATH`
- Ausführung im aktuellen Verzeichnis erzwingen: `./`

Beispiel: `./prog <Eingabe>`

Aufgabe 2

Schreiben Sie ein Programm, das die folgenden Aufgaben ausführt:

- ein Array vom Typ int mit 20 Elementen deklariert
- dieses Array mit den Quadratzahlen von 1 – 20 beschreibt
- den Inhalt des Arrays auf dem Bildschirm ausgibt, und zwar in der Form: `Die Zahl xxx zum Quadrat ist yyy`

Hinweise:

- Array deklarieren: `int zahlen[n];` n=Zahl der Elemente

- for-Schleife: `for(x=0; x<10;x++)`
  mit `x=0` erste Anweisung
  `x<10` Schleifenbedingung (muss erfüllt sein)
  `x++` Schleifenanweisung, wird am Ende der Schleife ausgeführt

Musterlösung

```
/* Die Quadratzahlen von 1 bis 20 ausgeben */

#include <stdio.h>

main()
  {
   int index;
   int zahlen[20];

   for(index=0; index<20; index++)
    {
     zahlen[index] = (index+1)*(index+1);
     printf("\n Die Zahl %d zum Quadrat",index+1);
     printf(" ist %d", zahlen[index]);
    }
  }
```

Aufgabe 3

Schreiben Sie ein Programm, das die folgenden Aufgaben ausführt:

- eine Datenstruktur für die Speicherung einer komplexen Zahl definiert und eine Variable von diesem Typ deklariert
- Realteil und Imaginärteil in die komplexe Zahl einliest
- den Inhalt des komplexen Zahl auf dem Bildschirm ausgibt, und zwar in der Form: `x + j y` oder `x – j y`

Hinweise:

- Datenstruktur definieren:

```
struct name
       {
        typ komponente
       };
```

- Fallunterscheidung: `if (Bedingung){...}else{...}`

Musterlösung (1)

```
/* Komplexe Zahlen einlesen und ausgeben */

#include <stdio.h>

struct komplex
  {
   int realteil;
   int imaginaerteil;
  };

main()
  {
   struct komplex zahl;
   printf("\n Bitte Realteil eingeben: ");
   scanf("%d",&zahl.realteil);
```

Musterlösung (2)

```
printf("\n Bitte Imaginärteil eingeben: ");
scanf("%d",&zahl.imaginaerteil);

printf("\n\n %d ", zahl.realteil);

if (zahl.imaginaerteil > 0)
  {
   printf("+ j %d",zahl.imaginaerteil);
  }
else if (zahl.imaginaerteil < 0)
  {
   printf("- j %d", -(zahl.imaginaerteil) );
  }
}
```