



Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich



Institut für  
Technische Informatik und  
Kommunikationsnetze

# Fachpraktikum 1 von 6: Systemprogrammierung in C

**Testatbedingungen:** Lösen der Aufgaben 1 bis 6  
**Zusatztestat:** keins

**Hinweis:** **Jeder arbeitet für sich**, Betreuer/Nachbar fragen erlaubt  
**Abgabe an Betreuer:** Programm(e) auf Konsole ausführen und zeigen,  
Vorhandene Fragen aus Text beantworten,  
Eigenen Quelltext erklären

**Communication Systems Group  
ETH Zurich**

**Ariane Keller** <ariane.keller@tik.ee.ethz.ch>  
**Daniel Borkmann** <daniel.borkmann@tik.ee.ethz.ch>

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Aufgaben</b>	<b>1</b>
2.1	Host/Netzadresse . . . . .	1
2.2	Kommandozeilenoption . . . . .	1
2.3	Substring . . . . .	2
2.4	Fibonacci . . . . .	2
2.5	Verkettete Liste . . . . .	2
2.6	Speicherfehler . . . . .	3

## 1 Einleitung

Das Ziel dieser Fachpraktikumsserie ist, das Verständnis der Betriebssystemkonzepte, wie sie in Technische Informatik II gelernt werden, zu vertiefen. Dazu offerieren wir sechs Fachpraktika die sich mit der Systemprogrammierung von Linux Rechnern befassen. In den Fachpraktikas wird die Programmiersprache C verwendet. Das erste Fachpraktika ist deshalb eine Einführung in die Programmiersprache C. Wenn Sie mit C schon vertraut sind, können Sie gleich mit den Aufgaben beginnen, sind Sie mit C noch nicht vertraut, können Sie sich zuerst das Begleitdokument zur Einführung in C ansehen. Lesen Sie sich dort die Kapitel 1 - 4 vollständig durch, sie erhalten Informationen über man pages, Syntax, das Kompilieren und mögliche Fehlermeldungen. In Kapitel 5 sind die Unterschiede von C zu Java und C++ erläutert, dieses Kapitel empfehlen wir Ihnen als Referenz für spätere Fachpraktikas.

Achtung, obwohl C und C++ sehr ähnlich sind, gibt es doch zahlreiche Unterschiede. Deshalb empfiehlt es sich, sich auch dann die Einführung in C anzusehen, wenn bereits gute C++ Kenntnisse vorhanden sind.

## 2 Aufgaben

### 2.1 Host/Netzadresse

In dieser Aufgabe sollen Sie aus einer gegebenen Host IP-Adresse die zugehörige Netzadresse bestimmen. Diese lässt sich mittels einer bitweisen AND Verknüpfung von Host IP und Subnetzmaske berechnen.

Verwenden Sie dazu die Vorlage `net.c`.

### 2.2 Kommandozeilenoption

Schreiben Sie ein kurzes Programm, welches beim Aufruf zwei Zahlen zwischen 0 und 9 als Parameter benötigt. Diese werden dann multipliziert und das Ergebnis soll auf der Konsole ausgegeben werden. Denken Sie daran die Eingabeparameter zu überprüfen.

Erstellen Sie dazu ein neues File.

Beispiel Programmaufruf:

```
./a.out 3 4
```

## 2.3 Substring

Programmieren Sie eine Funktion, die aus einem string einen substring herausliest. Die Funktionsdeklaration sieht folgendermassen aus:

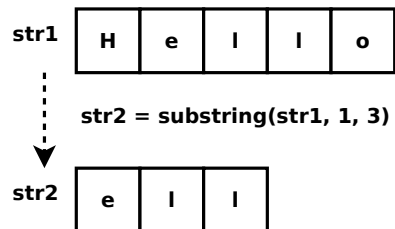
```
char *substring(const char *str, off_t off, size_t len)
```

An diese Funktion sollen drei Parameter übergeben werden: ein String `str`, ein Offset `off` und eine Länge `len`. Die Funktion soll aus dem übergebenen String `str` einen Substring herauskopieren und diesen an den Aufrufer zurückliefern. Der zurückgegebene String enthält damit nur noch einen Teil des ursprünglichen Strings beginnend bei der Position `off` mit der Länge `len`.

Weshalb wurden als Typ für die Argumente `off_t` und `size_t` anstatt zum Beispiel `int` gewählt?

Achten Sie bei Ihrer Implementation darauf, die Parameter auf ihre Gültigkeit zu überprüfen.

Ergänzen Sie die Vorlage `substr.c`.



## 2.4 Fibonacci

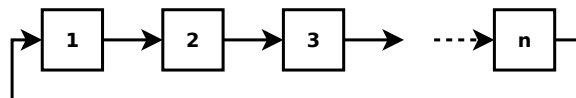
Schreiben Sie eine rekursive Funktion welche die Fibonacci-Folge berechnet. Geben Sie die 10. Fibonacci-Zahl aus.

Erstellen Sie dazu ein neues File.

$$f_n = \begin{cases} f_{n-1} + f_{n-2} & ,n \geq 2 \\ 1 & ,n = 1 \\ 0 & ,n = 0 \end{cases}$$

## 2.5 Verkettete Liste

Schreiben Sie ein Programm, welches eine einfach verkettete zirkuläre Liste erzeugt. Das heisst, das letzte Element muss wieder auf das erste Element zeigen. Ein Listenelement besteht aus einem `struct`, der einen Integer enthält und einen Pointer auf das nächste Element. Im Integer soll die Listenposition des Elementes gespeichert sein. Am Ende soll die Liste zwei Mal traversiert werden. Dabei soll jeweils der aktuelle Iterationsschritt sowie die Listenposition auf dem Terminal ausgegeben werden. Verwenden Sie für diese Aufgabe die zugehörige Vorlage `linked_list.c`.



## 2.6 Speicherfehler

Betrachten Sie das abgebildete Programm. Was erwarten Sie für eine Ausgabe? Kompilieren Sie das Programm mit dem folgenden Befehl:

```
gcc -w -o error error.c
```

Führen Sie dann das Programm aus und überlegen Sie sich, wie das Resultat zu stande kommt.

Ändern Sie die Funktion `f` so, dass das Resultat Ihrer Erwartung entspricht.

Verwenden Sie dazu die Vorlage `error.c`.

```
int *f(int a)
{
    int b = 2 * a;
    return &b;
}

int main(void)
{
    int *p4, *p8;
    p4 = f(4);
    p8 = f(8);
    printf("p4: %i / p8: %i\n", *p4, *p8);
}
```