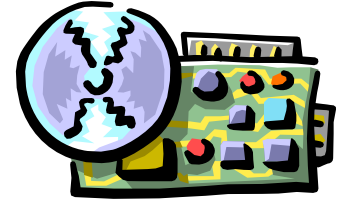




# TI III: Operating and Communication Systems



WS 2007/08  
Übungsblatt Nr. 1

Georg Wittenburg, M.Sc., AG Technische Informatik, Freie Universität Berlin

**Ausgabe am 19.10.2007 — Abgabe spätestens 2.11.2007, 10:00 Uhr**

Bitte bei der Abgabe beide Namen/Matr.Nr. der Mitglieder einer Gruppe, NUMMER DER ÜBUNG/TEILAUFGABE und DATUM auf den Lösungsblättern **nicht vergessen!** Darauf achten, dass die Lösungen beim richtigen Tutor/der richtigen Tutorin abgegeben werden.  
Achten Sie bei Programmieraufgaben außerdem darauf, dass diese im Linuxpool kompilierbar sind.  
**Zu spät abgegebene Lösungen werden nicht mehr berücksichtigt!**

---

## 1. Aufgabe: Begriffe (8 Punkte)

Beschreiben Sie jeden der folgenden Begriffe durch maximal 2 Sätze: Benutzerapplikation, Betriebssystem, Systemaufruf, Time Sharing Systeme, Microkernel, monolithischer Kernel, Instruction Set, Interrupt.

## 2. Aufgabe: Entwicklung von Betriebssystemen (6 Punkte)

Fassen Sie kurz in Stichpunkten die historische Entwicklung des Unix-Betriebssystems zusammen. Wann fanden Weiterentwicklungen in den Bereichen Speicherverwaltung, Mehrbenutzerbetrieb sowie der Anbindung an Computernetze statt.

## 3. Aufgabe: Protection Rings (4 Punkte)

Beschreiben Sie in kurzen Sätzen die Idee hinter den Protection Rings. Wieviel Ringe werden von gängigen Betriebssystemen zu welchem Zweck verwendet?

## 4. Aufgabe: Fifty/Fifty (8 Punkte)

Entscheiden Sie, ob folgenden Aussagen zutreffend oder nicht zutreffend sind, und begründen Sie Ihre Entscheidung:

- Ein Microkernel beschränkt sich auf grundlegende Speicherverwaltung und Kommunikation zwischen Prozessen.
- Im Kernelmodus muss das Betriebssystem darauf achten, dass die Speicherbereiche verschiedener Prozesse voneinander isoliert sind.
- Wenn Interrupt Service Routinen nebenläufig abgearbeitet werden, muss das Betriebssystem darauf achten, dass der Prozessor beim Timer-Interrupt nicht in den Benutzermodus wechselt.
- Bei Microkernen kommt es häufiger zu einem Kontextwechsel als bei monolithischen Kernen.

## 5. Aufgabe: Syscalls (8 Punkte)

Beschreiben Sie in wenigen Sätzen die vier in der Vorlesung vorgestellten Implementierungen von Systemaufrufen bei monolithischen Kernel und Microkernel in Hinsicht auf den Ablauf der Wechsel zwischen User- und Kernelspace.

## 6. Aufgabe: C Syntax (4 Punkte)

Kommentieren Sie das folgende Programm, ein Kommentar pro Zeile. Welches Spiel wird hier gespielt?

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <time.h>

#define BUFFER_SIZE 100

int main(int argc, char *argv[]) {
    int i, guess, target, goes, max = 100;
    char buffer[BUFFER_SIZE];

    srand((unsigned) time(NULL));    // Initialize random number generator.
    printf("Welcome!\n");
    for(i = 0; i < 8; i++) {
        printf("=");
    }
    printf("\n\n");

    do {
        goes = 0;
        target = rand() % max + 1;
        guess = 0;
        while(guess != target) {
            printf("Enter your guess: ");
            fgets(buffer, BUFFER_SIZE - 1, stdin);
            while(!sscanf(buffer, "%d", &guess)) {
                printf("Enter a number! ");
                fgets(buffer, BUFFER_SIZE - 1, stdin);
            }
            if(guess < target)
                printf("Too low! ");
            else if(guess > target)
                printf("Too high! ");
            ++goes;
        }
        printf("Well done, it was %d!\n", target);
        printf("You took %d goes.\n\n", goes);
        printf("Another go? (y/n) ");
        fgets(buffer, BUFFER_SIZE - 1, stdin);
    } while (buffer[0] == 'y' || buffer[0] == 'Y');

    printf("Goodbye!\n");
    return EXIT_SUCCESS;
}
```

## 7.Aufgabe: C Syntax (8 Punkte)

Das folgende Programm ist ein „Hello, world!“ in C:

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

int main(int argc, char *argv[]) {
    printf("Hello, world!\n");
    return EXIT_SUCCESS;
}
```

Verwenden Sie einen Text-Editor Ihrer Wahl, um das „Hello, world!“ Programm wie folgt zu modifizieren:

1. Geben Sie mit einer **for**-Schleife den String "Hello, world!\n" fünf mal hintereinander aus.
2. Geben Sie mit einer **while**-Schleife den String "Hello, world!\n " fünf mal hintereinander aus.
3. Geben Sie mit einer **do-while**-Schleife den String "Hello, world!\n " fünf mal hintereinander aus.
4. Geben Sie mit **if**-Anweisungen und einer Schleife Ihrer Wahl fünf mal abwechselnd die Strings "Hello, world!\n " und "Goodbye, world!\n" aus.
5. Geben Sie mit einer **switch**-Anweisung und einer Schleife Ihrer Wahl fünf mal abwechselnd die Strings "Hello, world!\n " und "Goodbye, world!\n" aus.
6. Geben Sie fünf mal in Abhängigkeit von einer Zufallszahl entweder "Hello, world!\n " oder "Goodbye, world!\n" aus.
7. Fragen Sie den Nutzer fünf mal "Hello, world?\n". Geben Sie in Abhängigkeit auf dessen Antwort (y/n) entweder "Hello\n" oder "Goodbye\n" aus.
8. Fragen Sie den Nutzer so lange "Hello, world?\n" bis dieser mit "n" antwortet. Schließen Sie das Programm dann mit einem "Goodbye, world!\n" ab. Solange er nicht mit "n" antwortet, sollte das Programm "Hello\n" ausgeben.

Benutzen Sie zur Lösung dieser Aufgaben den Quelltext aus Aufgabe 6 als Referenz.

## 8.Aufgabe: Kommandozeilenparameter (4 Punkte)

Die Funktion **main()** hat zwei Parameter:

- **int argc** - Anzahl der Parameter
- **char \*argv** - Array, das die Parameter als Strings enthält

Schreiben Sie ein Programm, das seine Kommandozeilenparameter in umgekehrter Reihenfolge ausgibt. Schlagen Sie hierzu auch mit dem Befehl

```
$ man 3 printf
```

nach, wie man einen String ausgibt.