3 Quest

3.1 Aufgabe

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

int main (int argc, char *argv[])
{
  int count;
  int i;
  FILE *f;

  if (argc < 2)
  {
    printf("Sie haben keine Parameter uebergeben!\n");
    exit(1);
  }
  else
  {
    for (i=1; i<argc; i++)
    {
      count = 0;
      f = fopen(argv[i], "r");
      if (f==NULL)
      {
        printf("%s %s %s\n", "Die Datei", argv[i], "konnte nicht geoeffnet werden!");
        exit(2);
      }
      else
      {
        while ( fgetc(f) != EOF )
            count++;
        fclose(f);
        printf("%s %s %d %s\n", argv[i], "enthielt", count, "Zeichen.");
      }
    }
    exit(0);
  }

}/*end of main*/
```

3.3 Aufgabe

Sprache	Typ	Befehl
C	Funktion	`fork()`
		`clone()`
Ada95	Deklaration	`task`

3.4 Aufgabe

```
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
  FILE *dz;
  int i;

  dz = fopen("test.txt", "w");
  if (dz==NULL)
  {
    printf("Die Datei test.txt konnte nicht geoeffnet werden!\n");
    exit(1);
  }
  else
  {
    switch(fork())
    {
      case -1:
        printf("fork-Fehler!\n");
        exit(1);
      case 0:
        for (i=0; i<10; i++)
        {
          fprintf(dz, "Kindprozess: %2d \n", i);
          fflush(NULL);
        }
        exit(0);
      default:
        for (i=0; i<10; i++)
        {
          fprintf(dz, "Elternprozess: %2d \n", i);
          fflush(NULL);
        }
        fclose(dz);
    }
    exit(0);
  }
}
```

3.5 Aufgabe

Ein Prozess ist ein Programm in Ausführung, inklusive Prozessraum. Jeder Prozess hat eine Ausf'hrungsrichtung und einen Ausführungspfad. Man kann mehrere Threads in einem Prozess führen. Dies ist vorteilhaft, da die Handhabung von Threads einfacher ist, ein Grund wieso sie auch als "leichtgewichtige Prozesse" bezeichnet werden. Insbesondere sind Threads innerhalb eines Prozesses ebenfalls unabhängig voneinander.
Auch das System kann einen Wechsel von Threads schneller handhaben als einen von Prozessen, unter anderem da Threads keine tatsächlichen pysikalischen Ressourcen zugeteilt bekommen. Ein Thread hat jedoch einen Befehlszähler und einen Stack für die Speicherung von Variablen innerhalb eines Prozesses zugewiesen.
Man spricht von "Multithreading", wenn mehrere Threads in einem Prozess (pseudoparallel) laufen. Man beachte, dass tatsächlich jedoch jeweils nur ein Thread auf der CPU auf einmal laufen kann.