REAL TIME SYSTEMS

Exercise 3: PIPES & SIGNALS

TASKS

M.Sc. Michael Rethfeldt

1 Introduction to pipes

a. See the given program source code aufgabe1.c
b. Compile it and explain its behavior!
c. Check the program by executing it with the file argument: aufgaben\testfile!

```
//////////////////////////////////////////////////////////////////////////////
// Course:                      Real Time Systems
// Lecturer:                    Dr.-Ing. Frank Golatowski
// Exercise instructor:         M.Sc. Michael Rethfeldt
// Exercise:                    3
// Task:                        1
// Name:                        aufgabe1.c
// Description:                 ?
//////////////////////////////////////////////////////////////////////////////
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <ctype.h>

void hex_print(FILE *dz, char *s)
{
        int fd[2];

        if (pipe(fd) != 0) {
                fprintf(stderr, "Error while creating pipe!\n");
                exit(1);
        } else {
                printf("Hex print of %s\n\n",s);
                switch(fork()) {
                case -1:{       fprintf(stderr, "fork() failed\n");
                                exit(1);
                         }
                case  0:{       int i=1;
                                unsigned char read_char;

                                close(fd[1]);
                                while (read(fd[0], &read_char, 1) > 0) {
                                        printf(" %02x", read_char);
                                        if (++i > 16) {
                                                printf("\n");
                                                i=1;
                                        }
                                }
                                printf("\n");
                                close(fd[0]);
                                exit(0);
                        }
                default:{       unsigned char c;
                                int status;

                                close(fd[0]);
                                while (fread(&c,1,1,dz) > 0)
                                        write(fd[1], &c, 1);
                                close(fd[1]);
                                wait(&status);
                        }
                }
        }
}

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

        if (argc < 2) {
               fprintf(stderr, "Call: aufgabe1 <filename>\n");
               exit(1);
        }
        for (i=1; i<argc; i++) {
               if ((dz=fopen(argv[i],"rb")) == NULL ) {
                      fprintf(stderr,"Can't open file %s\n", argv[i]);
                      exit(1);
               } else {
                      hex_print(dz,argv[i]);
                      fclose(dz);
               }
        }
}
```

Fig. 1: aufgabe1.c

2 Introduction to signals

a. Inform yourself about signals and signal handling.
b. What are signals needed for?
c. What signals do exist? Give some examples! How are they handled?
d. What signal-related functions are important in UNIX/Linux?

3 Signal handler

a. Write a program that creates a son process! The son prints its PID every second and sleeps in between. After ten seconds the son shall terminate automatically. The father process shall always ignore the SIGINT signal, wait for the termination of the son, and print the reason (status) of the son's termination. Only the son shall be quit by a keyboard input of Ctrl+C.
Hint: You can trigger an acoustic output by using printf("%c\n", '\7').

4 Reader and writer

a. In the program aufgabe1.c a father and son process exchanged data via a pipe. Now modify the program behavior as follows:

- The father creates two son processes that shall exchange data via a pipe.
- After creating the writer, the father shall close its writer side of the pipe (fd[1]).
- After creating the reader, the father shall close its reader side of the pipe (fd[0]).
  After that, the father shall wait for the termination of both sons.
- The writer first closes the reader side of its pipe (fd[0]).
- The reader first closes the writer side of its pipe (fd[1]).

After that, both son processes exchange data via the pipe. The writer sends data to the reader. The reader converts the data and prints out the results.