M.Sc. Michael Rethfeldt

1 Recap: Combination of fork() and exec()

a. See the source codes given in Fig. 1 & 2. Compile the programs for father and son!
b. Think about the progression of the father process! Check your considerations by executing the program!
c. Explain the function calls execl() und wait()!
d. When does the father print „wait status: …“?

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

int main(void)
{     int status;
      pid_t fork_pid;
      if ( (fork_pid=fork() ) == 0 )
      {     execl("./son", "son", NULL);
            printf("execl() failed!\n");
            exit(3);
      }
      else if (fork_pid == -1)
      {     printf("fork() failed!\n");
            exit(2);
      }
      wait(&status);   // wait for termination of son
      printf("wait status: 0x%x | 0x%x | 0x%x |\n", status,
                  (status>>8) & 0xff, status & 0xff);
      return 0;
}
```

Fig. 1: Father Process

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

int main(void)
{
      printf("I am the son.\n");
      return 0;
}
```

Fig. 2: Son Process

2 Cascaded fork() call

a. Analyze the program aufgabe2.c given in Fig. 3 and explain the behavior!

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

int main(void)
{
  int var;
  printf("PID(Father)=%d, var=%d\n", getpid(), var);
  if ( fork() == -1 )
  {
    fprintf( stderr, "fork() failed!\n" );
  }
  else
  {
    var++;
    printf("PID=%d, var=%d\n", getpid(), var);
    if ( fork() == -1 )
    {
        fprintf(stderr, "Error at 2nd fork()!\n");
    }
    else
    {
       var++;
       printf("PID=%d, var=%d\n", getpid(), var);
    }
  }
  return 0;
}
```

Fig. 3: aufgabe2.c

3 Arbitrary number of processes

a. Write a program that creates multiple son processes! The number of processes shall be passed as command line argument at program start! The father process may be terminated after all son processes have finished their job. Every son process shall sleep for a random value of time. Moreover, every process shall print something before going to sleep and after awakening.
Keep the final source code, you will need it again in one of the next exercises!

Hints:

srand(time(NULL)); will not work well since the timer value is the same for all processes. What would be a better solution?