Responsible lecturer: Illés, Zoltán associate professor
Media & Educational Informatics Department

Parent-child processes

Goal: We are to learn how to create a new process from code and what are the main features of them. We are getting to know exec function family - how we can replace a process image with a new process image. We learn the difference between exec and system functions.

We learn: *fork – creates a new child process (include unistd.h);getpid,getppid – gives back PID number; wait,waitpid – waits for the end of child process(es) (include sys/types.h, sys/wait.h); execv – replace the actual process with a program (include unistd.h); system – executes a command (include stdlib.h)*

Tasks

1. Write a program which creates a child process! Simply write out „Hello world" after creating the child process! (*How many times do you see „Hello world" on the screen?*)

```
pid_t fork(void);
// result – the PID number of the child process
```

2. Modify the program and call fork function twice! (*How many times do you see „Hello world" on the screen? Why? What would happen if you tried to use fork inside a for cycle? Let you think!*)

3. Write a C program in which parent process does another thing (writes out „I am the parent") than the child process (writes out „I am the child")! At the end of the parent process wait for the end of the child process! (*Only in parent process you can see the value of the child process PID number (it is greater than 0)! In child process it is 0! There is an error if the value is smaller than 0! You should wait for the end of the child process to avoid „zombie" processes!)*

```
pid_t wait(int *status);
// waits for the end of child process and gives back the state of it into status
```

4. Write a C program which creates 2 child processes! (*Be careful, second fork should be called only inside parent process not to create 2^2 processes! How to create 3 or more child processes?*)

5. Try to guess what would happen if you executed the undergiven code? *(How many child processes you get? Will there be any 0 values written out by printf? If it is so than explain it why? Complete the code to avoid zombie processes!)*

```
pid_t getpid(void);
// the pid number of the child

pid_t getppid(void);
// the pid number of the parent
```

6. Modify the program so that child processes should execute only a custom function and nothing else! *(Please, check what is written out on the screen in this case!)*

7. Write a C program which writes out N times a given text. The number and the text is given by command line arguments. Write another C program which creates a child process and replace the child with the formerly written program. (*You have to use one of the exec function-family. Pleace, check what happens after the execution of the exec function! The child is replaced so nothing is available (printf) after the execution of exec.)*

```
int execv(const char *path, char *const argv[]);
// path – command, argv – command with arguments
```

Financed from the financial support ELTE won from the Higher Education Restructuring Fund of the Hungarian Government

8. Modify the program an change exec function to system call. *(Check whether you see the result of printf or not! Try to explain the result!)*

9. Write a mini shell with exec! *(In parent you have to read in commands from the keyboard in a cycle. For each command creates a child process and inside it call exec.)*