Jacobs University Bremen

Assignment #10

Student name: *Ivan Kabadzhov*

Course: *Introduction to Computer Science*
Date: *December, 2018*

Problem 10.1: fork system call

a)

```c
#include <unistd.h>

int main(int argc, char *argv[])
{
    for (; argc > 1; argc--) {
      if (0 == fork()) {
        (void) fork();
      }
    }
    return 0;
}
```

Note that the argument counter `argc` is always greater than 0, as it includes the program that was executed to get the process running.

- `./foo` $\Rightarrow$ `argc`$= 1 \Rightarrow$ not entering the `for` loop. Child processes $= 0$.
- `./foo a` $\Rightarrow$ `argc`$= 2 \Rightarrow$ forking inside the `if` statement and also in `(void)fork()`. Child processes $= 2$.
- `./foo a b` $\Rightarrow$ `argc`$= 3 \Rightarrow$ forking inside the `if` statement and also in `(void)fork()`. Child processes $= 8$. Then `argc` decrements and all new processes are forked.
- `./foo a b c` $\Rightarrow$ `argc`$= 4 \Rightarrow$ Following the same logic. Child processes $= 26$.

Therefore observe that:

argc	child processes
1	$0 = 3^{1-1} - 1$
2	$2 = 3^{2-1} - 1$
3	$8 = 3^{3-1} - 1$
4	$26 = 3^{4-1} - 1$

Hence the number of resulting child processes is $3^{\texttt{argc}-1} - 1$.

- `./foo a b c d` $\Rightarrow$ `argc`$= 5 \Rightarrow$ Following the same logic. Child processes $= 80$.

b)

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

int main(int argc, char *argv[])
{
    for (; argc > 1; argc--) //condition of the loop remains the same
    {
        pid_t child_pid; //creating a child process
        child_pid=fork();
        if(child_pid > 0) //restating the previous condition
        {
            while(1) {}
        }
    }
    exit(0); //creating the zombie
    return 0;
}
```

Problem 10.2: recursive directory tree walk

```
#include <dirent.h> //for directory traversing
#include <stdio.h>
#include <string.h>

void directory_content(char * path)
{
    DIR * d = opendir(path); //open directory path
    if(d == NULL)
        return;
    struct dirent * dir; //directory entries
    while ((dir = readdir(d)) != NULL)
    {
        if(dir-> d_type != DT_DIR)
        {
            char directory_path[100];//arbitrary large
            sprintf(directory_path, "%s/%s", path, dir->d_name); //concatenation
            printf("%s\n",directory_path);
        }
        else
        {

            if(dir -> d_type == DT_DIR && strcmp(dir->d_name, ".") != 0
                    && strcmp(dir->d_name, "..") != 0) // if directory
            {
                char d_path[100];
                sprintf(d_path, "%s/%s", path, dir->d_name);
```

```
                 //store the data as a string
                for(int i = 0; i < strlen(d_path); i++)
                {
                    //print names in the path as consecutive strings
                    printf("%c",d_path[i]);
                }
                printf("\n");
                directory_content(d_path); //recursion
            }
        }
    }

    closedir(d); //close directory path
}


int main(int argc, char **argv)
{
    if (argc>1)
        directory_content(argv[1]);//call the function
    if (argc==1)//as we saw in 1.a. argc is always greater than 0
    {//thus, start counting from 1
        directory_content(".");
    }

    return(0);
}

/*three hierarchy:
|---a---|
|   |   |
x   b   y
   / \
  c   z */
```