Lab 5 Ex 1 - SDP - Davide Gallitelli S241521

The goal of this exercise is to find out which are the system calls involved when running the command:

```
wc < test.txt | grep 1
```

The sequence starts with 22 SYS_READ calls, as many as the characters contained in the above mentioned command.

```
5 - SYS_read [x22]
[...]
```

Then, the shell performs a SYS_FORK call, so that its child child will execute the command. After that, the shell executes a SYS_WAIT, in order to wait for child termination. The SYS_sbrk is the system call version of the *malloc()* function, which allocates memory to the newly created process.

```
1 - SYS_fork
3 - SYS_wait
12 - SYS_sbrk
```

The shell first child can now execute the command. This command is actually made of two commands piped: the child executes a SYS_pipe system call to create a nameless pipe to allow for interprocess communication.

```
4 - SYS_pipe
```

The following system calls are explained in the file *sh.c*, which contains information about shell execution. When a *pipe* instruction, or related system call, is received, the shell forks twice, creating two children, which will be referred to in the following lines as *CH1* and *CH2*, to handle the two commands, *wc < test.txt* and *grep 1*. Finally, it calls for a SYS_wait in order to wait for children to complete execution.

```
1 - SYS_fork
1 - SYS_fork
3 - SYS_wait
```

The two children created both perform four system calls: CLOSE, DUP, CLOSE, CLOSE.

```
// First children
21 - SYS_close
```

```
10 - SYS_dup
21 - SYS_close
21 - SYS_close
// Second children
21 - SYS_close
10 - SYS_dup
21 - SYS_close
21 - SYS_close
```

CH1, the child responsible for *wc*, performs a SYS_exec syscall to change its shell to the *wc* command. Then, it performs a SYS_close syscall to close the file descriptor related to *stdout*, because it will write on the pipe, and the one related to the read from the pipe.

```
21 - SYS_close
21 - SYS_close
7 - SYS_exec
```

Also the CH2, which will execute *grep*, performs SYS_exec and two SYS_close, closing the file descriptor related to *stdin*, as it will read the output from the pipe, and the one for writing on the pipe, because it will write on *stdout*.

```
21 - SYS_close
21 - SYS_close
7 - SYS_exec
```

The first step of the *wc* command is to open the file *test.txt*, which is done by a SYS_open syscall. It reads 512 bytes, receiving 31 bytes (one per character). Then it reads again from the file, returning 0 bytes meaning that it reached the EOF.

```
15 - SYS_open
// read outputs of the wordcount (1) (4) (31)
5 - SYS_read    // read 512 bytes
5 - SYS_read    // read 0 bytes - EOF
```

Then, CH1 can write on the pipe the 8 bytes generated as output from the *wc* command, then exits.

```
16 - SYS_write
16 - SYS_write
16 - SYS_write
16 - SYS_write
16 - SYS_write
16 - SYS_write
```

```
16 - SYS_write
16 - SYS_write
2 - SYS_exit
```

After being awaken by the child termination, it waits for the other child to die.

```
3 - SYS_wait
```

CH2 can now read from the pipe by means of a SYS_read syscall, and writes the output to *stdout*. Then reads again from the pipe to look for the end of the pipe, and it exits.

```
5 - SYS_read
16 - SYS_write  // [ 1  4 31]
5 - SYS_read
2 - SYS_exit
```

The shell child now exits and returns control to the shell.

```
2 - SYS_exit
```