Operating Systems
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CO20-320202
Date: 2016-11-30
Deadline: 2016-12-06

Problem Sheet #6

Problem 6.1: *open files and file updates or meta data changes* (1+1+1+1+1 = 5 points)

```
/*
 * catloop.c --
 */

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

int
main(int argc, char *argv[])
{
    int fd;
    pid_t pid;

    if (argc != 2) {
        fprintf(stderr, "catloop: missing 'file' argument\n");
        return EXIT_FAILURE;
    }

    fd = open(argv[1], O_RDONLY);
    if (fd == -1) {
        perror("catloop: open");
        return EXIT_FAILURE;
    }

    while (1) {
        pid = fork();
        if (pid == -1) {
            perror("catloop: fork");
            (void) close(fd);
            return EXIT_FAILURE;
        }
        if (pid == 0) {
            char c;
            (void) lseek(fd, 0, SEEK_SET);
            while (read(fd, &c, 1) == 1) {
                write(STDOUT_FILENO, &c, 1);
            }
            (void) close(fd);
            return EXIT_SUCCESS;
        }
        (void) waitpid(pid, NULL, 0);
        sleep(1);
    }

    (void) close(fd);
    return EXIT_SUCCESS;
}
```

Save the source code shown above into the file `catloop.c`. Compile the C code to produce the

executable `catloop` and afterwards execute the following shell commands:

```
$ rm -f foo
$ touch foo
$ ./catloop foo &
$ echo -n "hello " > foo
```

Now answer the following questions (always with the same initial setup):

a) What is the program doing?

b) What happens if you append content to the file `foo` while `catloop` is running?

```
$ echo "world" >> foo
```

c) What happens if you truncate the file `foo` while `catloop` is running?

```
$ truncate -s 0 foo
```

d) What happens if you change the permissions of the file `foo` while `catloop` is running?

```
$ chmod 0 foo
$ ls -l foo
```

e) What happens if you remove the file `foo` while `catloop` is running?

```
$ rm -f foo
```

Solution:

a) The program opens the file provided as a command line argument for reading and then forks every 2 seconds a child process, which seeks to the beginning of the file and then copies the content of the file to the standard output. Note that the parent keeps the file open.

b) Changes to the file's content become immediately visible when the file is read again.

c) Similarly, file truncation becomes immediately visible. Note that this may lead to incomplete copies of the file's content.

d) Changes of file access permissions do not affect open files. Access permissions are checked when a file is opened but not anymore afterwards.

e) The file disappears from the directory immediately but the file's content remains accessible as long as there are open file descriptors referring to it (which is the case as long as the `catloop` parent process is running).

Problem 6.2: *file system permissions* (1+1+1+1+1 = 5 points)

Unix file system objects have basic permissions associated with (i) the file owner, (ii) the file's group members, and (iii) everybody else with access to the file system. Please answer the following questions:

1. Who has which access permissions for the file `foo`?

```
$ ls -l foo
-rwxrw-r-- 1 schoenw adm 0 Nov 30 14:53 foo
```

2. Who has which access permissions for the directory `bar`?

```
$ ls -ld bar
drwx-wx--- 2 schoenw adm 4096 Nov 30 14:56 bar
```

3. Can a member of the group `adm` (who is different from `schoenw`) read the content of the directory bar? Can a member of the group `adm` (who is different from `schoenw`) create a file in the director bar?

4. A regular user (with a umask of 0022) executes the following shell command. What are the file permissions of the file that is created and who is the owner of the file?

```
$ rm -f world
$ sudo echo hello > world
```

5. What is the meaning of the following access permissions?

```
$ ls -l /usr/bin/sudo
-rwsr-xr-x 1 root root 157760 Jan 11  2016 /usr/bin/sudo
```

Solution:

a) The owner of the file (`schoenw`) has read, write, and execute permissions. The members of the group `adm` have read and write permissions. Everybody else has read permissions.

b) The owner of the directory (`schoenw`) has read, write, and execute permissions. Execute permissions on a directly mean that it is possible to traverse the directory. The members of the group `adm` have write and execute permissions. Everybody else has no access permissions at all.

c) A member of the group `adm` (who is different from `schoenw`) cannot read the content of the directory `bar`. However, such a user is able to create files in the directory `bar` due to the write and execute permissions.

d) The `sudo` command causes the `echo` program to be executed with `root` permissions. However, the output redirection is setup by the shell before the `sudo` command is executed and hence the new file will be owned by the user running the shell. The permissions of the file will be as follows:

```
$ ls -l world
-rw-r--r-- 1 schoenw schoenw 6 Nov 30 15:16 world
```

e) The `s`-bit set on the user access permissions indicates that the file will be executed with the permissions of the file owner (in this case `root`) instead of the permissions of the user invoking one of the `exec()` system calls. Note that everybody has the permissions to read and execute the `sudo` command.