Operating Systems (234123) - Spring 2016
Home Assignment 4 - Dry

Due Date:	Tuesday, 21.6.2016, 12:30p.m.
Teaching assistant in charge:	Matthias Bonne

Postponements can be authorized **only** by Arthur, the TA in charge. If you need a postponement, please contact him directly.

Important: The Q&A for the exercise will take place at a public forum Piazza only. Critical updates about the HW will be published in pinned notes in the Piazza forum. These notes are mandatory and it is your responsibility to be updated. A number of guidelines to use the forum:

- Read previous Q&A carefully before asking the question; repeated questions will probably go without answers.
- Be polite, remember that course staff does this as a service for the students.
- You are not allowed to post any kind of solution and/or source code in the forum as a hint for other students; in case you feel that you have to discuss such a matter, please come to the reception hour.
- When posting questions regarding hw4, put them in the hw4 folder.

Dry part submission instructions:

1. Please submit the dry part to the electronic submission of the dry part on the course website.
2. The dry part submission must contain a single `dry.pdf` file containing the following:
   (a) The first page should contain the details about the submitters - name, ID number and email address.
   (b) Your answers to the dry part questions.
3. Only typed submissions will be accepted. Scanned handwritten submissions will not be accepted.
4. Only `PDF` format will be accepted.
5. You do not need to submit anything in the course cell.

שאלה 1 (50 נקודות)

קיראו על הפונקציות poll() ו־sigprocmask() וענו על השאלות הבאות. בסעיפים 4־2 הניחו שכל הסיגנלים חסומים בתחילת הריצה של הפונקציה, ושמותקן הנדלר עבור הסיגנל SIGUSR1.

1. (10 נקודות) השלימו את החלק החסר בפונקציה הבאה:

```
int double_wait (int fd1, int fd2)
{
        /* ... */


        return poll (fds, 2, -1);
}
```

הפונקציה מקבלת שני file descriptors כפרמטרים, ומחכה עד שלפחות אחד מהם יהיה מוכן לקריאה (כלומר, ניתן לבצע עליו read() ללא המתנה).

2. (10 נקודות) כעת יש לממש את הפונקציה:

```
int double_wait_safe (int fd1, int fd2);
```

הפונקציה זהה לפונקציה double_wait() מסעיף 1, פרט לכך שהיא חוזרת מיד כשמתקבל סיגנל SIGUSR1.

סטודנט טוען שלפי ההגדרה, הפונקציה poll() חוזרת מיד כשמתקבל סיגנל כלשהו, ולכן אין צורך לשנות את הפונקציה מסעיף 1. במילים אחרות, הסטודנט מציע את המימוש הבא:

```
int double_wait_safe (int fd1, int fd2)
{
        return double_wait (fd1, fd2);
}
```

הסבירו בקצרה (לכל היותר שני משפטים) למה מימוש זה לא עובד כנדרש.

3. (10 נקודות) כדי להתגבר על הבעייה, הסטודנט מציע לשחרר את הסיגנל SIGUSR1 מיד לפני הקריאה ל־poll(). לשם כך, הסטודנט מציע להחליף את השורה:

```
        return poll (fds, 2, -1);
```

בשורה:

```
        return xpoll (fds, 2, -1);
```

ולהגדיר את הפונקציה xpoll() כך:

```
int xpoll (struct pollfd *fds, nfds_t nfds, int timeout)
{
        int retval;
        sigset_t origmask, newmask;

        /* unblock SIGUSR1 */
        sigemptyset (&newmask);
        sigaddset (&newmask, SIGUSR1);
        sigprocmask (SIG_UNBLOCK, &newmask, &origmask);

        /* call poll() with SIGUSR1 unblocked */
        retval = poll (fds, nfds, timeout);

        /* restore the original signal mask and return */
        sigprocmask (SIG_SETMASK, &origmask, NULL);

        return retval;
}
```

תארו מצב שבו המימוש המוצע לא יעבוד כנדרש.

4. (20 נקודות) ממשו את הפונקציה double_wait_safe(). לצורך הפתרון מותר להשתמש בכל פונקציה שממומשת כחלק מהספריה הסטנדרטית בלינוקס.

שאלה 2 (50 נקודות)

שאלה זו עוסקת בחלק השני של התרגיל.

1. (10 נקודות) תארו בעייה שהיתה נוצרת אם הפקודה ioctl(RNDCLEARPOOL) לא היתה דורשת הרשאות root.

2. (40 נקודות) סטודנט שכח להשתמש ב־copy_to_user() כשמימש את המתודה read() של הקובץ. פרט לכך הניחו שהמימוש עובד כנדרש, והניחו שהערכים שמוחזרים על־ידי read() מפולגים באופן אחיד על־פני כל הערכים האפשריים.

(א) (10 נקודות) כיתבו פונקציה שהופכת את התהליך הנוכחי לתהליך FIFO עם priority מקסימלי. הניחו שלתהליך הנוכחי יש הרשאות root.

(ב) (20 נקודות) כיתבו פונקציה שהופכת את התהליך הנוכחי לתהליך FIFO עם priority מקסימלי, בהנחה שלתהליך הנוכחי <u>אין</u> הרשאות root. לשם כך הניחו שקיים קובץ בשם /dev/srandom שמייצג את ההתקן שמימש הסטודנט הנ"ל ושניתן לקריאה על־ידי התהליך הנוכחי, והניחו שהערך current->cap_effective נמצא בכתובת __ADDR__[1].

(ג) (10 נקודות) נניח שתהליך ללא הרשאות root קורא לפונקציה שהגדרתם בסעיף ב. מה ההסתברות שהפונקציה תבצע לכל היותר קריאה אחת ל־read()?

(ד) (לא להגשה) מהי תוחלת מספר הקריאות ל־read() שמתבצעות במהלך ריצת הפונקציה שמימשתם בסעיף ב, בהנחה שלתהליך הקורא אין הרשאות root?

[1] השדה cap_effective ב־task_struct מכיל את סט ההרשאות של תהליך (thread) מסויים. הפעולה הנדרשת כאן (הפיכת התהליך הנוכחי לתהליך real-time) היא אפשרית אם ורק אם ביט 23 בשדה זה דלוק. הניחו שגודל השדה הוא 32 ביט.