Homework 3 Dry

Dan Sdeor dansdeor@campus.technion.ac.il 209509181

Levi Hurvitz levihorvitz@campus.technion.ac.il 313511602

Due Date: 12.1.2023

Teaching assistant in charge: Mousa Arraf
209509181
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're 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 hw3 , put them in the hw3 folder

Only the TA in charge can authorize postponements. In case you need a postponement, fill in this form -
https://forms.office.com/r/7hJXUUXA3T

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.
6. When you submit, **retain your confirmation code and a copy of the PDF**, in case of technical failure. It is **the only valid proof** of your submission.

יש לנמק כל תשובה אלא אם במפורש נאמר אחרת, תשובות ללא נימוק לא יתקבלו.

שאלה 1 - Networking - תקשורת (50 נק')

1. **(4 נק')** באיזו אבסטרקציה מרכזית של לינוקס משתמשים כאשר מתקשרים באמצעות ממשק ה-sockets?

בלינוקס Everything is a file. כלומר לאחר שיצרנו socket עם פרמטרים ספציפיים לאופן התקשורת שלנו מול הhost אליו אנחנו מתקשרים (לא חייב להיות מחשב פיזי, יכול להיות גם תהליך). השימוש בsocket הוא כמו שימוש בfile descriptor סטנדרטי שדיברנו עליו עד עכשיו. כלומר אני יכול לכתוב לsocket על ידי קריאה לwrite (שידור נתונים), לקרוא מהsocket עם read (קליטת נתונים) או לסגור את הsocket עם close כי הsocket הוא fd.

2. **(6 נק')** ציינו את שתי הבעיות העיקריות בהעברת נתונים ברשת וכיצד מתמודדים איתן.

בעיה 1: פקטות שנשלחו מעל הרשת יכולות שלא להגיע (buffer של switch יכול להתמלא למשל מפקטות אחרות והswitch לא יקבל פקטות חדשות).

בעיה 2: פקטות שנשלחו על גבי הרשת יכולות שלא להגיע בסדר הנכון, למשל הפקטה השנייה תגיע לפני הפקטה הראשונה בשידור. הסיבה לכך שיכולים להיות מספר מסלולים ברשת שבהם חבילה יכולה לעבור ולכן כל נתב יכול להחליט שהוא מעביר פקטות במסלולים שונים.

כיצד מתמודדים איתן: ישנם פרוטוקולים רבים שפותרים את הבעיה הזו אבל אחד הרעיונות המרכזיים שממומש בTCP הוא להגדיר לכל header של tcp segment שני שדות, SEQ וACK. כאשר סגמנט tcp נשלח ברשת, מיד בצד השולח מתחיל לרוץ timer. אם עבר הtimeout שהשולח מגדיר עבור הסגמנט ולא קיבלנו סגמנט מהצד המקבל עם ack ששווה לseq+segment size אז מבחינתנו הסגמנט לא יתקבל אצל הצד המקבל ונשלח את הסגמנט שוב. אם הצד המקבל, מקבל סגמנט עם seq שונה ממה שהוא ציפה לקבל אז הוא שומר את אותו סגמנט בבאפר שלו (במידה ולא התקבל סגמנט עם seq כזה בעבר) וממשיך לשלוח ack עבור הסגמנט שהוא מצפה לקבל (ברגע שיש לו סדרה של segments מסודרים אז הוא יכול להעביר אותם לאפליקציה ולנקות קצת את הבאפר). ככה למעשה הפרוטוקול מוודא שסגמטים יגיעו ואם הם לא מגיעים בסדר הנכון אז שמישהו ידאג לסדר אותם.

3. **(12 נק')** מלאו את הטבלה שלמטה כך שתסביר את ההבדל בין כתובת MAC, כתובת IP, ופורט.

	תפקיד המספר	כיצד הוא מתקבל
MAC Address	משמש פרוטוקולים שמוגדרים בdata link layer כדי לזהות התקנים פיזיים ברשת. כאשר חבילה עוברת על גבי הרשת היא יכולה לעבור דרך התקני רשת רבים. תפקיד הכתובת הוא להבטיח מעבר בין התקן אחד לשני בצורה פרטנית.	היצרן שמייצר את התקן הרשת, צורב באופן ישיר למשל בrom של המוצר הוא בsection כלשהוא של הfirmware של הchipset ככה שלא אמורים להיות שני התקנים בעלי אותה כתובת mac (אחריות היצרן לוודא את זה).
IP Address	להבטיח נכונות לוגית של הרשת. כאשר מחשב רוצה לשלוח חבילה למחשב שלא נמצא בLAN שלו. כתובת mac לא תעזור לנו. אבל התקן יכול לשלוח את החבילה לפי הrouting table שלו לdefault gateway אם לא נמצא חוק מתאים יותר. כך החבילה יכולה לעבור מעל נתבים מתאימים עד שהיא תגיע לרשת שמתאימה לsubnet המתאים ובסופו של דבר לנקודת קצה. במילים אחרות, מבנה הרשת לא אמור לקבוע האם שליחת חבילה תתאפשר ואם היא לא מתאפשרת נקודתית אז מחפשים מי ההתקן המתאים ביותר לקבל אותה להמשך שידורה.	ישנם שתי דרכים להגדיר כתובת של התקן רשת בLAN. הדרך הראשונה היא להגדיר את הכתובת באופן ידני. כלומר static ip address. צריך לוודא בשיטה הזאת שהip של הdefault gateway הוא הip של הנתב של אותה רשת לוקאלית ושהsubnet mask של ההתקן מוגדר כמו צריך,שיש לנו כתובת ip של השרת dns כדי שנוכל להמיר domain names לip ולבסוף שאין התקן קיים ברשת עם הכתובת ip הזאת. אם זה לא המצב לא נוכל לדעת לאיזה התקן פיזי צריך לשלוח את המידע בdata link layer. הדרך השנייה היא לעשות את זה באופן אוטומטי באמצעות הפרוטוקול dhcp. כאשר ההתקן מתחבר לרשת אז הוא שולח בקשת dhcp request בbroadcast לכל הרשת. השרת dhcp (בבתים פרטיים הוא בדרך כלל גם הנתב אבל זה לא מחייב) שולח dhcp response את כל הפרטים שהוא היה צריך לקנפג ידנית עם הip שיהיה לו ברשת. בפועל מדובר ב4 שלבים שבהם ההתקן שולח discovery בboradcast כדי להגיד לשרת שהוא צריך כתובת ip. השרת ישלח offer עם פרטי הרשת. אם ההתקן בוחר לקבל אותם אז הוא ישלח request שיכיל את הנתונים ולסיום השרת ישלח acknowledge שהבקשה התקבלה
Port	לדעת לשלוח חבילה מהתקן להתקן שכן ברשת או לישות לוגית שלא בהכרח נמצאת איתי באותו רשת לא מספיקה. מה אם יש על אותה ישות מספר תהליכים שמבקשים שירות מישות לוגית כלשהי או מספקים שירות לישויות לוגיות אחרות. כדי שנוכל לדעת לאיזה תהליך (שרת כלשהו) המידע הנשלח מיועד נצטרך להגדיר מזהה ייחודי: dst port וכדי שנוכל להחזיר תשובה לתהליך	עבור listening port. אותו dst port שמשמש את שכבה 4 לזהות שירותים עבור השרת אליו מפנים את החבילה. אותו port יוגדר על ידי המשתמש כשהוא קורא לbind עם הsocket שהוא יצר לשם הגדרת הport שמשויך לשירות. לעומת זאת כאשר תהליך כלשהו רוצה לתקשר עם שרת אז בהקמת הsocket עבור הלקוח. לרוב מערכת ההפעלה תקצה את הPort לאותו socket באופן זמני מתוך טווח של Portים עד שהfd יסגר. אותו port ישמש בתור הsrc port של החבילות שהלקוח שולח

	הספציפי ששלח את ההודעה נצטרך מזהה ייחודי שהוא הsrc port. לסיכום, עם src ip addr,dst ip addr,src port, dst port,transport protocol אנו יכולים לבקש או לתת לישות לוגית ברשת שירות באופן יחודיי, ip מאפשר לזהות את הישות והport את הישרות שהיא מבקשת	

4. **(20 נק')**

א. וּוילֶם, מנכ"ל של חברת הייטק נחשבת, חושש שספקית האינטרנט שלו תאזין לתקשורת שלו ותמכור מידע רגיש למתחרים ולכן הוא שוקל להצפין את ה-payload של שכבה 2 אשר המחשב שלו משדר (מבלי לתאם את הדבר עם אף גורם). באיזה שלב בדרכה של החבילה אל היעד ניתקל בבעיה לראשונה?

במצב הזה החבילה תיתקע אצל הראוטר שיקבל את החבילה מהמחשב של ווילם (אם החבילה מיודעת לנקודת קצה מחוץ לרשת המקומית כמו במקרה של ווילם). אותו ראוטר יוציא את הdata שהוא ip packet בשכבה שלוש מהethernet frame בשכבה 2. לאחר מכן כשהוא ינסה לפענח את הip header, ויגלה שהוא הושחת מכיוון שלאחר ההצפנה שדה הchecksum של הheader כבר לא תקף ביחס לchecksum שחושב על הheader ולכן הוא יחליט לזרוק את החבילה מחוסר תקינות.

ב. אם ווילם היה מצפין את ה-payload של שכבה 3 (במקום של 2), באיזה שלב בדרך של החבילה אל יעדה ניתקל בבעיה לראשונה?

כעת ווילם מצפין את הpayload של שכבה 3, כלומר את שכבה 4. במצב כזה נצטרך להתייחס לשתי רשתות שונות:

רשת בה הנתבים מממשים את מנגנון NAT: זו רשת אינטרנט טיפוסית, במצב כזה כאשר התקן הרשת של המחשב של ווילם ירצה ישדר את החבילה והיא תתקבל אצל הראוטר. אותו ראוטר ישלוף את התוכן מתוך שכבה 2 שלא מוצפן, תוכן זה הוא פקטת ip שמכילה את כתובת הip של הידע וכתובת הip של המחשב של ווילם אותו הראוטר יחליף לכתובת הIp שלו אבל הוא צריך גם להחליף את port המקור של החבילה לport מקור אחר שיבחר על ידי הראוטר. שדה הport נמצא בשכבה 4 שהושחתה כתוצאה מההצפנה (שדה הchecksum כבר לא זהה לchecksum שחושב על ידי הראוטר). לכן הראוטר יבחר שלא להעביר את החבילה הלאה ויזרוק אותה.

רשת בה הנתבים לא מממשים את מנגנון NAT: במציאות, יכולה להיות רשת off-grid של אוניברסיטה או חברת הייטק קטנה שמחלקות/פקולטות שונות בה הן LANים שונים שלכל אחד יש subnet משלו ולכן אין בעיה לפחות ברמה הפנימית של חלוקת כתובות ip. אבל מצב לא ריאליסטי ברשת גלובלית. במצב הזה אם ווילם ירצה לשלוח חבילה למחשב קצה ברשת הארגונית כל הנתבים/גשרי ביניים במעבר של החבילה יפתחו אותה עד לשכבה 2/3 בהתאמה (כלומר המעבר הלוגי של החבילה לפי טבלאות ניתוב או ברמת הlink בין מחשבים לא נפגעת). לכן החבילה תגיע למחשב קצה ברשת. אותו מחשב יפתח אותה עד לשכבה 4 וכאשר יגיע לשכבה זו יזרוק אותה כמו שהוסבר בפסקה הקודמת מאחר והיא ניזוקה כתוצאה מההצפנה.

5. **(8 נק')** לווילם יש קופסא של בזק בבית שמשתמשת בשני מנגנונים עיקריים המאפשרים לכל התקן פרטי ברשת הביתית שלו לקבל כתובת IP שמאפשרת תקשורת אינטרנט. תארו את שני המנגנונים בקצרה.

DHCP- פרוטוקול שמאפשר לתחנות קצה להתחבר לרשת מקומית ולקבל מידע חיוני על הרשת כדי לתקשר עם מכשירי קצה אחרים ברחבי האינטרנט. המחשב מקבל מידע חיוני כמו כתובת הIp שלו, כתובת הIp של הנתב, כתובת הip של השרת dns, הsubnet mask של הlan שיאפשר לדעת שהחבילה צריכה לעבור לנתב עם היעד לא ברשת.
NAT- מספר כתובות הip שניתן להקצות לכל המכשירים בעולם הוא קטן ביחס למספר המכשירים הקיימים ולכן כדי לפתור את הבעיה (another level of indirection) הוחלט להקצות טווח כתובות פרטיות שאותם ראוטרים יחלקו למכשירים ברשת הלוקאלית וכאשר אותה חבילה תשלח ליעד שנמצא מחוץ לרשת, הראוטר שיקבל את החבילה ידאג לשנות בשכבה 3 את כתובת הip של השולח (לכתובת הip שלו) ובשכבה 4 את הport של השולח (לפורט אחר בדרך כלל גבוה יותר) כאילו הוא שלח את החבילה ודואג להחליף את המזהים בחזרה כשהשרת שולח חבילה לראוטר. ככה בעצם אפשר לחלק את אותה כתובת Ip פרטית להרבה מאוד מחשבים כל אחד ברשת פרטית משלו.

שאלה 2 - סינכרון (50 נק')

המצאת המושג "פקולטה נחשבת" החמירה את הסכסוך בין הסטודנטים במדמ"ח ובהנדסת חשמל, ולכן הוגדר כי כאשר סטודנט מאחת הפקולטות רוצה להיכנס לחדר מסויים עליו לציית לכלל הבא: אם יש סטודנטים מפקולטה אחרת בחדר אזי אסור לסטודנט להיכנס ועליו להמתין עד שיעזבו (לעומת זאת, מספר סטודנטים מאותה פקולטה יכולים לשהות בחדר באותו הזמן).

סמני נכון / לא נכון (אין צורך להסביר):

1. (3 נק') יכולים להיות שני סטודנטים מפקולטות שונות באותו חדר במקביל: נכון / לא נכון
2. (3 נק') יכולים להיות שני סטודנטים מפקולטות זהות בחדר במקביל: נכון / לא נכון
3. (3 נק') סטודנטי פקולטה אחת עלולים להרעיב (כניסת) סטודנטי פקולטה אחרת: נכון / לא נכון

בסעיפים הבאים מוצג קוד למימוש כניסה ויציאה של סטודנטים אל ומחדר מסוים, כאשר נתון כי:

- כל חוט מייצג סטודנט.
- בכניסה לחדר הסטודנט קורא ל onArrival(int faculty), שמקבלת את פקולטת הסטודנט.
- ביציאה מהחדר הסטודנט קורא ל onLeave(int faculty) שמקבלת את פקולטת הסטודנט.
- הערכים 0 ו-1 של faculty מייצגים את הפקולטה להנדסת חשמל ומדמ"ח, בהתאמה.
- (הניחו שאמצעי הסנכרון עברו אתחול תקין והתעלמו מבעיות קומפילציה אם ישנן, שכן מטרת השאלה אינה לבדוק שגיאות אתחול/תחביר).

```
#include <pthread.h>

int students = 0;
mutex_t global;

void onLeave(int faculty) {
        mutex_lock(&global);
        students--;
        mutex_unlock(&global);
}
```

```
void onArrival(int faculty) {
        mutex_lock(&global);
        while (students > 0) {

        mutex_unlock(&global);
                sleep(10);
                mutex_lock(&global);
        }
        students++;
        mutex_unlock(&global);
}
```

1. (12 נק') בהתייחס לקוד הנ"ל, הקיפי את כל התשובות הנכונות (עשויה להיות יותר מאחת).
עבור כל תשובה שהקפת, תארי דוגמת הרצה המובילה לתשובה זו.

a. קיימת בעיית נכונות עקב race condition למשאבים משותפים.
b. קיימת בעיית DeadLock / Livelock בקוד.
c. הקוד משתמש ב-Busy Wait שפוגע בנצילות המעבד.
d. הקוד מפר את כלל הכניסה לחדר (שהוגדר בתחילת השאלה).

נימוק:

e. הקוד משתמש ב-Busy Wait שפוגע בנצילות המעבד:

סטודנט מפקולטה 0 נכנס לחדר על ידי קריאה לOnArrival נועל את הmutex, רואה שstudents==0, מוותר על הלולאה מגדיל את הstudents ב1 ומשחרר את המנעול. נכנס סטודנט שני מפקולטה 0 לחדר וקורא לOnArrival, בגלל שstudents==1 אז הוא יכנס ללולאה, ינעל את הmutex, יקרא לsleep(10), ישחרר את הmutex, יבדוק שstudents>0 ושוב יכנס ללולאה. המעבד ימשיך להריץ פקודות מכונה של החוט הזה שוב ושוב למרות שעוד לא יתקיים התנאי שיאפשר לחוט להיכנס לחדר (להגדיל את הstudents) התנאי שיכול להתקיים רק אם הסטודנט הראשון יצא. צריך לשים לב שלא מדובר בפגיעה של כלל הכניסה לחדר כי למרות שהוא מתיר לשני סטודנטים או יותר מאותה פקולטה לשהות בחדר, הקוד הנ"ל מחליט שרק סטודנט אחד יכול לשהות בתורו וזה לא נחשב פגיעה בחוק אלא רק ניצול לא טוב שלו.

המימוש של כניסה ויציאה שונה כך שישתמש במשתני תנאי:

```
int students[2] = {0};      // 2 counters
cond_t conds[2];            // 2 condition variables
mutex_t global;
void onArrival(int faculty) {
        mutex_lock(&global);
        int other = faculty ? 0 : 1;
        while(students[other] > 0)
                cond_wait(&conds[faculty] , &global);
        students[faculty]++;
        mutex_unlock(&global);
}
void onLeave(int faculty) {
        mutex_lock(&global);
        students[faculty]--;
        int other = faculty ? 0 : 1;
        cond_broadcast(&conds[other]);
        mutex_unlock(&global);
}
```

אך דני (עתודאי במדמ"ח) טען שקוד זה גורם לחוטים להתעורר שלא לצורך ומיד לחזור למצב המתנה.

1. (8 נק') הסבירי את טענתו של דני באמצעות דוגמת ריצה קונקרטית.

נניח ששני חוטים מאותה פקולטה (פקולטה 0) נכנסים לonArrival. אחד מהם ינעל את המנעול, יראה שלא צריך להיכנס ללולאה כי אין חוטים של הפקולטה האחרת בחדר יגדיל את מונה הפקולטה שלו וישחרר את המנעול לחוט השני שיעשה את אותו הדבר. כעת מגיעים עשרה חוטים מפקולטה 1. אחד כל פעם יספיק לפני האחרים לנעול את המנעול ויכנס לקטע קוד קריטי. יראה שיש סטודנטים מפקולטה אחרת (students[0] == 2) ויכנס לתור המתנה בעקבות cond_wait. לאחר שכל סטודנטים פקולטה 1 ממתינים, סטודנט מפקולטה 0 קורא לonLeave. הוא נועל את המנעול מקטין את מספר הסטודנטים מפקולטה 0 ב1 ומעיר את כל הסטודנטים מפקולטה 1 ומשחרר את המנעול. כל אחד מסטודנטי פקולטה 1 ינעל את המנעול בתורו, יראה שstudents[0] == 1, יכנס ללולאה, ישחר את המנעול ויחזור לתור המתנה בעקבות cond_wait. כלומר, נוצר מצב שסתם הערנו סטודנטים שלא לצורך.

1. (8 נק') כיצד ניתן לתקן את הבעיה שהציג דני בסעיף הקודם?

רק כאשר בשלב שחוט עושה onLeave, נכנס לקטע קוד קריטי, מקטין את מונה הסטודנטים מהפקולטה שלו ורואה שהוא התאפס, כלומר שהוא הסטודנט האחרון מאותה פקולטה אז הוא רשאי לקרוא לcond_broadcast כי במצב זה באמת ישנה אפשרות ממשית לסטודנטים מהפקולטה האחרת לצאת מהלולאה לאחר התעוררות.

דני ניסה לשפר עוד את יעילות הקוד והחליט להשתמש בשני מנעולים: מנעול ראשון בעבור סטודנטים הנכנסים לחדר, ומנעול שני בעבור סטודנטים היוצאים מהחדר. להלן המימוש החדש (השינויים בקוד מודגשים):

```
int students[2] = {0};              // 2 counters
cond_t conds[2];                    // 2 condition variables
mutex_t m_arrival, m_leave;         // there are *2* locks now
void onArrival(int faculty){
        mutex_lock(&m_arrival);
        int other = faculty ? 0 : 1;
        while(students[other] > 0)
                cond_wait(&conds[faculty] , &m_arrival);
        int tmp = students[faculty];
        students[faculty] = tmp + 1;
        mutex_unlock(&m_arrival);
}
void onLeave(int faculty){
        mutex_lock(&m_leave);
        int tmp = students[faculty];
        students[faculty] = tmp – 1;
        int other = faculty ? 0 : 1;
        cond_broadcast(&conds[other]);
        mutex_unlock(&m_leave);
}
```

1. (13 נק') בהתייחס לקוד הנ"ל, הקיפי את כל התשובות הנכונות (עשויה להיות יותר מאחת).
עבור כל תשובה שהקפת, תארי דוגמת הרצה המובילה לתשובה זו.

a. יתכנו 2 סטודנטים מפקולטות שונות בתוך החדר ביחד, עקב race condition למשאב משותף.
b. יתכן סטודנט שלא נכנס לחדר למרות כלל הכניסה שמתיר זאת, עקב race condition למשאב משותף.
c. קיימת בעיית DeadLock / Livelock בקוד.
d. סיגנלים עלולים ללכת לאיבוד.

נימוק:
e. יתכנו 2 סטודנטים מפקולטות שונות בתוך החדר ביחד, עקב race condition למשאב משותף:
סטודנט 1 מפקולטה 0 נכנס לחדר, נועל את m_arrival, עובר את הלולאה כי אין אף אחד בחדר מפקולטה 1,מגדיל את המונה של פקולטה 0, משחרר את המנעול ויוצא מהפונקציה. סטודנט 1 מפקולטה 0 מעוניין לצאת מהחדר, הוא קורא לonLeave, נועל את m_leave, מגיע לשורה 16 כאשר tmp==1 וקורה החלפת הקשר . כעת נכנס סטודנט 2 מפקולטה 0 לחדר הוא עובר את כל השלבים (בגלל שזה לא אותו מנעול אז הוא יכול להיכנס לקטע הקוד הקריטי של onArrival) ומסיים את onArrival. לאחר שסטודנט 1 מפקולטה 0 חוזר מהמתנה הוא כותב למונה את tmp-1 כלומר המונה של פקולטה 0 הוא 0 למרות שיש שם סטודנט 2 מפקולטה 0. כעת סטודנט 1 מפקולטה 1 יכול להיכנס לonArrival ולסיים מבלי להיכנס לwait_cond. כלומר שני סטודנטים מפקולטות שונות נמצאים באותו חדר.

f. יתכן סטודנט שלא נכנס לחדר למרות כלל הכניסה שמתיר זאת, עקב race condition למשאב משותף.

סטודנט 1 מפקולטה 0 נכנס לחדר, נועל את m_arrival, עובר את הלולאה כי אין אף אחד בחדר מפקולטה 1,מגדיל את המונה של פקולטה 0, משחרר את המנעול ויוצא מהפונקציה. כעת נכנס סטודנט 2 מפקולטה 0 לחדר הוא עובר את כל השלבים עד שהוא מגיע לשורה 10 כאשר tmp==1 ואז מתרחשת החלפת הקשר. סטודנט 1 מפקולטה 0 מעוניין לצאת מהחדר, הוא קורא לonLeave, נועל את m_leave (אין בעיה כי זה לא אותו מנעול), משחרר אותו ויוצא מהפונקציה. לאחר שסטודנט 2 מפקולטה 0 חוזר מהמתנה הוא כותב למונה את tmp+1 כלומר כותב 2 למונה של פקולטה 0 ויוצא מהפונקציה לאחר שחרור המנעול. המונה של פקולטה 0 הוא 2 למרות שיש שם רק סטודנט אחד מפקולטה 0. אותו סטודנט יוצא מהחדר ועכשיו יש 0 סטודנטים בחדר מפקולטה 0 והמונה של הסטודנטים מפקולטה 0 הוא 1. כאשר סטודנט מפקולטה 1 ינסה להיכנס לחדר הוא לא יצליח ויכנס להמתנה כי המונה של פקולטה 0 גדול מ0 למרות שאין שם סטודנטים.

g. סיגנלים עלולים ללכת לאיבוד.

אם סטודנט 1 מפקולטה 0 נכנס לחדר ואז יוצא אז cond_broadcast(&conds[other]); בפונקציה onLeave פשוט תשלח סיגנל למרות שאין אף חוט אחר שמחכה לאותו סיגנל (נניח שרק סטודנט 1 מפקולטה 0 נכנס ויצא). הסיגנל פשוט ילך לאיבוד והפעולה חסרת השפעה.