

מבחן במערכות הפעלה

ד"ר סיון טולדו, מועד ב', סמסטר ב' תש"ס

הוראות כלליות

- חומר העזר המותר היחיד הוא דף בגודל A4 כתוב בכתב היד שלכם (כלומר בכתב היד של התלמיד הניגש למבחן).
- חובה לצרף את דף העזר למחברת המבחן! מבחן שלא יצורף אליו הדף יפסל.
- משך המבחן שלוש שעות.
- בהצלחה!

שאלה א'

תאר (י) בפרוטרוט את תרגום הכתובת הכתובת הוירטואלית `0x007a34bc` לכתובת פיזית. המחשב משתמש בדפים בגודל 8192 בתים, ולכל בית בזכרון יש כתובת. כתובות וירטואליות הן בנות 32 סיביות, וכתובות פיזיות בנות 36 סיביות. למחשב טבלת דפים היררכית בת שתי רמות שבה לטבלה ברמה העליונה יש 1024 כניסות. (1) מה גודל הטבלאות ברמה השנייה (מה מספר הכניסות)? (2) תאר (י) בדיוק את אלגוריתם התרגום וכיצד ניבנית הכתובת הפיזית. אם יש יותר מדרך אחת לביצוע התרגום, תאר (י) את כל הדרכים, ובאיזה מקרים יעשה שימוש בכל אחת מהדרכים.

שאלה ב'

הוטל עליך להסב תוכנית שמשמשת בחוטים, מנעולים, ואירועים (events או condition variables) ויוניקס לחלונות (Windows).

הבעיה היא שקריאות המערכת הקשורות לאירועים אינן פועלות באותה דרך ביוניקס ובחלונות. כזכור, ביוניקס הקריאה `signal` (ליתר דיוק `pthread_cond_signal(c)`) מעירה בדיוק חוט אחד מבין החוטים שממתינים על האירוע, כלומר שקראו ל-`wait` לפני שבוצעה הקריאה ל-`signal` ושעדיין מחכים. אם אין חוטים שמחכים, אין ל-`signal` שום השפעה. כמו כן, הקריאה `wait(c,m)` משחררת את המנעול `m` ומכניסה את החוט להתנה באופן אטומי.

בחלונות לעומת זאת הפונקציה `signal(c)` מסמנת שהאירוע קרה, ואם אין חוטים שממתינים, מערכת ההפעלה זוכרת שהאירוע קרה. אם בהמשך חוט קורא ל-`wait`, מערכת ההפעלה תשחרר אותו מייד מהמתנה. כמו כן, אירוע בחלונות משחרר מהמתנה את כל החוטים הממתינים ואת החוטים שיכנסו בעתיד להתנה, וזאת עד שקוראים לקריאת המערכת `reset(c)`. חוט שיקרא ל-`wait` לאחר שבוצע `reset` ימתין.

הבדל נוסף הוא שבחלונות לקריאה `wait` יש רק ארגומנט אחד, אירוע, והיא אינה משחררת שום מנעול.

עליך לממש את ארבע הקריאות `unix_lock(m)`, `unix_unlock(m)`, `unix_signal(c)`, ו-`unix_wait(c,m)` כאשר `m` היא כתובת של מנעול ו-`c` היא כתובת של אירוע, בעזרת חמש הקריאות `win_lock(m)`, `win_unlock(m)`, `win_signal(c)`, `win_reset(c)`, ו-`win_wait(c)`. בקריאות המערכת של חלונות

הארגומנטים הם עצמים של חלונות; הארגומנטים של ארבעת הפונקציות שעליך לממש הם כתובות של struct-ים שעליך להגדיר (כך שייצגו עצמים עם התנהגות יוניקס).
הפונקציות שעליך לממש חייבות להתנהג ממש כמו הפונקציות המקוריות ביוניקס, וזאת כדי שהתכנית תרוץ ללא שגיאות.
אין צורך לדאוג להגינות.

שאלה ג'

1. מתי צריך להשתמש בקריאת המערכת readlink? האם אפשר להשתמש ב--open ו--read במקומה?
2. למה מגבים נתונים? תאר בקצרה שלושה תרחישים שונים שיצריכו שימוש במידע מגיבוי ואת סוג הגיבוי הדרוש לכל תרחיש.
3. מתי ולמה קריאת המערכת execv אינה חוזרת לתוכנית הקוראת?

שאלה ד'

ידיד העובד כתוכניתן במוסד פנה אליך בשאלה בנושא מערכות הפעלה. תוכנית שכתב ושמצעת בעיקר גישות לקבצים רצה לאט. הידיד מודע לכך שכדאי לבצע קלט/פלט בבלוקים גדולים ולכן ארגן את התוכנית כך שכל הגישות הן לבלוקים בגודל 1MBytes ומעלה של הקבצים. למרות זאת, קצב העברת הנתונים בפועל הוא בערך 0.5MBytes/second ואף למטה מזה. מטעמי סודיות הידיד לא היה מוכן לגלות את סוג המחשב או מערכת ההפעלה, אך היה מוכן לספר שמבחינת החומרה מדובר במחשב אישי רגיל למדי.

1. מה הבעיה? (נמק'י).
2. הצע (י) שני פתרונות לבעיה.
3. לאיזה קצב העברת נתונים בערך יכול ידיך לצפות אם יישם את הפתרונות שהצעת?

שאלה ה'

1. כתבת תוכנית יוניקס/לינוקס המשתמשת במספר חוטים לפתרון בעיות חישוביות כבדות. לקוח המשתמש בתכנית צלצל אליך וסיפר שהתוכנית נתקעה לקראת סוף ריצה, לאחר 6 ימי ריצה. המשתמש סיפר שהתוכנית עדיין תקועה והוא לא הורג אותה בתקווה שהיא תשתחרר ותסיים את עבודתה. אתה משער שמדובר ב--deadlock. כיצד תוכל לברר בודאות מה הבעיה (במהירות!)? האם הרצות נוספות על אותו הקלט יתקעו גם הן?
2. תוכנית יוניקס אחרת שכתבת היא שרת http. השרת יוצר חוט חדש עבור כל בקשת התחברות. בחרת בארגון כזה של התוכנית מכיון שכך מתאפשר מתן שירות למספר לקוחות גדול בו זמנית (כאשר יש מספר רב של לקוחות כמובן שזמן התגובה עולה, אך כולם ממשיכים לקבל שירות רצוף), וגם מכיון שכך מתאפשר לך לשמור בזכרון האמיתי דפים (=קבצים) שניגשו אליהם לאחרונה. חוט שמטפל בבקשה יכול לבדוק במבנה נתונים משותף האם הקובץ שהוא צריך לספק נימצא כבר בזכרון ואם כן, אין צורך לקרוא אותו מהדיסק. אם לא, החוט קורא את הקובץ ומשאיר אותו בזכרון עבור חוטים אחרים שאולי יצטרכו אותו. (התוכנית שומרת לכל היותר 64MBytes של קבצים בזכרון

על ידי מחיקה של קבצים שלא נגשו אליהם זמן רב ממבנה הנתונים בזכרון). אחד המשתמשים של התוכנה שלך הוא ספק אינטרנט שבדרך כלל השרתים שלו רצים ללא הפסקה במשך שבועות או חודשים. המשתמש הזה דיווח שהתוכנית שלך גורמת לשרתים שלו לקרוס (מערכת ההפעלה עפה) לאחר שהיא רצה זמן מה, כשבועיים. לא נראה שהדבר נגרם מדליפת זכרון כיון שמערכת ההפעלה מדווחת שהתוכנית משתמשת בכמות קבועה של זכרון, כ-100MBytes. כמו כן בבדיקה קפדנית שערכת לא מצאת בתכנית שום דליפת זיכרון (למשל malloc ללא free מתאים), וכמות דפי האינטרנט שהתכנית שומרת בזכרון מוגבלת תמיד ל-64MBytes. המשתמש מאיים שהוא ירכוש תוכנה מתחרה אם הבעיה לא תפתר. מה יכולה להיות הבעיה? כיצד ניתן לתקן את הבעיה?

שאלה ו'

1. גלישה באינטרנט מרשת התקשורת האוניברסיטאית מתבצעת רק דרך proxy. כיצד לדעתך מונעת האוניברסיטה התחברות ישירה לאתרי אינטרנט (כלומר שלא דרך ה-proxy)? מדוע לדעתך מרשה האוניברסיטה גלישה אך ורק דרך ה-proxy?

שאלה ז'

1. מה לדעתך עושה התכנית הבאה? מה תפקיד השורה הראשונה? מה עושה השורה השלישית (set x ואיך?)

```
#!/bin/csh
foreach f (*.c)
  set x = 'ls -l $f | cut -c 30-41'
  set y = 'ls -l $f | cut -c 56-80'
  if ( $x > 2000 ) then
    echo $y
  endif
end
```