לינוקס (ושאר דמויי־יוניקס) למת"מניקים

אוהד לוצקי ובועז גולדשטיין

29 ביולי 2005

תוכן עניינים

1 למה לעזאזל אנחנו רוצים סביבת עבודה דמויית יוניקס?

הקורס מת"מ משלב עבודה אינטנסיבית בסביבת יוניקס. באופן טבעי, למרבית האנשים אין שרתי Sparc בבית, או כל מחשב אחר שמריץ מערכת יוניקס קלאסית. במת"מ ההמלצה היא לעבוד מול T2 באמצעות Telnet, אשר לה מספר חסרונות:

- השימוש ב־Telnet כפי שהוא מתאפשר ב־T2 לוקה בבעיות אבטחה חמורות – כל המידע (כולל שם המשתמש והסיסמא) נשלח בצורה לא מוצפנת. מסיבה זו Telnet גם חסום ברשתות רבות, דוגמת רשת המחשבים הניידים בטאוב.
- תוכנת Telnet אינה מאפשרת בקלות שימוש בתוכנות גרפיות, כגון Emacs (במצב הגרפי שלו) או `ddd`.
- פרוטוקול Telnet אינו מאפשר העברת קבצים, ודורש שימוש בפרוטוקול בעייתי אחר – FTP. פרוטוקול זה הינו מגושם (שימוש בפורטים רבים) ולא־מאובטח בדיוק כמו Telnet.

את כל הבעיות שצויינו אפשר לפתור בקלות ע"י שימוש ב־SSH[1]. זהו פרוטוקול מאובטח ועדכני אשר מחליף את Telnet ואת FTP. הוא מאפשר העברת קבצים (באמצעות תוכנה נפרדת, אך דרך אותו פרוטוקול מאובטח ויעיל), ובהינתן שרת X (יוסבר בהמשך), מקל מאוד על הרצת תוכנות גרפיות דרך הרשת. ישנם לקוחות SSH רבים, והקל ביותר לשימוש בחלונות – אשר מותקן ברוב מחשבי הטכניון – הוא הלקוח של Tectia, שניתן להשיג באתר `http://www.ssh.com`.

אך השימוש ב־SSH משאיר בעיה אחת מרכזית בעבודה מול T2 – מהירות. T2 הינו שרת מרוחק, והעבודה מולו מחוץ לטכניון היא איטית למדי כאשר משתמשים בתוכנות גרפיות. אך גרוע מכך – T2 הינו מחשב מאוד עמוס, ובזמני העומס סטודנטים רבים מקמפלים ומריצים תוכנות שעושות שימוש כבד בארבעת מעבדיו – והדבר נוטה להביאו לברכיו.[2]

מהסיבות שצויינו, רצוי להתארגן על סביבת עבודה דמויית־יונקס על מחשב פרטי יחסית – כלומר, מחשב שמשמש פחות מ־100 משתמשים בכל רגע נתון.

2 למי שרוצה לעבוד מחלונות

מאחר ורוב הסטודנטים מריצים את מערכת ההפעלה "חלונות" בביתם, רצוי שתהיה להם אפשרות להשתמש בה בתוכנות שרלוונטיות לקורס. רבים נוטים להמליץ למת"מניקים להשתמש דווקא בקומפיילר Microsoft Visual C++. ישנם מספר חסרונות לשיטה זו:

- הקומפיילר של Microsoft Visual C++ אינו דומה ל־`gcc`, במיוחד כשמדובר במצב הקפדני (`-pedantic`). מסיבה זו ישנו סיכוי מאוד גבוה שקומפילציה על T2 תניב שגיאות ואזהרות רבות יחסית אשר Visual C++ לא יתריע מפניהן.
- פתרון זה אינו נותן מענה לכלים יוניקסיים נוספים שמלמדים בקורס, כגון `csh`. בעוד שחלק מהכלים (`make`, למשל) כן מופיעים בסביבת פיתוח זו, הם פועלים בצורה שונה מאוד מהגירסאות היוניקסיות המקורית שלהן. `nmake`, למשל, פועל בצורה שונה מ־GNU `make`, ואין להשתמש בו במסגרת הקורס.

---

[1] Secure Shell

[2] יתרון למשתמשי ה־TX – אף אחד לא נוגע בו, ויש לו 16 (!) מעבדים.

למרבה המזל, חברת Red Hat (מפיצת לינוקס) יצרה מערכת כלים בשם Cygwin. מדובר בסביבה דמויית־יוניקס לחלונות. חברת Microsoft מייצרת חבילה דומה בשם Services for unix, שמכילה את כל הכלים הדרושים למת"מ, אך לא הרבה מעבר – חסרים כלים מודרניים ונוחים, למשל valgrind. מסיבה זו נדבר כאן על Cygwin, אשר כוללת תוכנות שמלמדים להשתמש בהן בקורס כגון gcc, csh, gdb, ddd, Emacs, vim, cd, בנוסף לשרת X מתוחכם.

2.1 Cygwin

סביבת העבודה Cygwin קלה מאוד להתקנה, ונותנת סביבה מאוד דומה ללינוקס (גם מערכת דמויית־יוניקס), ולמעשה תהיה זהה ב־95% ל־T2. הבדלים משמעותיים שרצוי לשים לב אליהם בין Cygwin לבין t2, לינוקס, ושאר היוניקסים.

- תוכנות ב־Cygwin מקבלות סיומת exe. (כי אלו בכל־זאת תוכנות בחלונות)
- לפעמים אין חשיבות ל־cAsE של אותיות.
- קבצי הטקסט הם DOS-Formatted, שוב – מאחר ומדובר בחלונות. מה זה אומר? שבסוף כל שורה יש שני תווים: CR ולאחריו LF[3]. ביוניקס, לעומת זאת, ישנו רק תו אחד בסוף כל שורה – LF. gcc יודע להתגבר על ההבדל הזה תמיד, csh בדרך הכלל – ומסיבה זו ברוב עורכי הטקסט יש אפשרות להעביר בין מצב unix ומצב DOS.

ההתקנה היא פשוטה יחסית. יש ללכת לאתר http://www.cygwin.com, ולהוריד את תוכנת ההתקנה. זהו קובץ קטן שיוריד את סביבת היוניקס המלאה מהאינטרנט. ניתן להוריד את כל הסביבה, לצרוב לדיסק, ולהתקין במקום אחר. יש לשים לב בעת בחירת המראה (ניתן לבחור מאיזה אתר מראה רוצים להוריד) ששרתי FTP חסומים מהטכניון, ולכן יש לבחור בשרת HTTP. תוכנות שרצוי לסמן מתוך רשימת התוכנות להתקנה: xorg, gcc, tcsh, openssh, ddd ועורך הטקסט האהוב עליכם (נניח VIM או Emacs).

2.2 שרת X?!

בתרגולים הראשונים במת"מ נוצר הרושם אצל הסטודנטים שיוניקס היא סביבה טקסטואלית לחלוטין. אם להיות מדוייקים לחלוטין, ליוניקס אין כלל וכלל ממשק משתמש – לא גרפי ולא טקסטואלי. הממשק הטקסטואלי בו אנו משתמשים נקרא מעטפת או Shell (למשל sh, csh או bash). כמו־כן קיים גם ממשק גרפי, שנקרא X או X11, ויש לו מימושים רבים, כגון XFree86 ו־Xorg.

ממשק ה־X, בדומה לממשק הטקסטואלי, בנוי עם מחשבה לכיוון של שימוש מרוחק, כלומר דרך רשת. בתכנונו עלתה הדרישה שלא יהיה שום הבדל בין עבודה גרפית על המחשב המקומי לבין עבודה בלפטופ עם חיבור אינטרנט לווייני ממדבר סהרה. לתכונה זו קוראים Network Transparency. למעשה X[4] או X11 הוא פרוטוקול תעבורה גרפית ברשת.

מפחיד, נכון? לא ממש. אז בואו קצת נדבר תכל'ס. מה אתם צריכים כדי להריץ תוכנות גרפיות ישירות מ־T2 על המחשב שלכם? ראשית, אתם צריכים *שרת X*. זוהי תוכנה אשר מקבלת הוראות ציור למסך מתוכנות יוניקס, ולמעשה מציגה בפניכם את הממשק הגרפי שלהן. ברגע שסביבת העבודה דמויית־היוניקס שלכם תותקן, ירוץ שרת X כלשהו.

---

[3] אלו הם ה־M^־ים הידועים לשמצה

[4] למה X? כי לפני זה קראו לו W (עבור Windowing), ואז שיפרו אותו מאוד והחליטו לקדם אותו באות.

בלינוקס: שרת ה־X כבר רץ, והסביבה הגרפית שאתם משתמשים בה למעשה משתמשת בשרת ה־X. אין צורך לבצע פעולה מיוחדת כדי להפעיל אותו.[5]

בסביבת Cygwin: כאשר אתם מפעילים את קיצור הדרך Cygwin שנוצר, הוא למעשה מציג חלון Shell ללא שרת X. הקישו את הפקודה הבאה:

```
$ startx
```

כעת יופעל שרת X (שימו לב לצלמית באיזור ה־Tray), ויורץ על־גביו `xterm` – זהו חלון פקודות שמודע לסביבה הגרפית, ומתוכו תוכלו להריץ תוכנות גרפיות, על־אף שהוא צהוב[6].

תוכנה נוספת שתצטרכו היא SSH. כאמור, ל־SSH תכונה מאוד שימושית שמנתבת אוטומטית את התעבורה הגרפית אל המחשב שלכם. כל התיאוריה המעניינת משלוש הפסקאות האחרונות מסתכמת בדוגמה הבאה: כיצד להריץ את `ddd` ישירות מ־T2 בסביבה דמויית־יוניקס:

```
$ ssh -X USERNAME@t2.technion.ac.il
password:
> ddd
```

האופציה `X-` היא שמפעילה את המאגיה השחורה (טוב, לא ממש – רק משתנה סביבה וקצת ניתוב מידע) שגורמת ל־`ddd` להופיע על המחשב שלכם. משתמשי תוכנות SSH אחרות (גרפיות) – חפשו בתפריטים של התוכנה את האופציה של X Forwarding.

2.3 העברת קבצים ל־ ומ־T2

ניתן להשתמש ב־Secure File Transfer Client כדי לבצע פעולה זו, אבל זה לא בשורת הפקודה אז זה לא כיף. בנוסף, זה גם לא קשור לחומר שמלמדים בקורס. אז הנה שתי שיטות שכן קשורות.

2.3.1 תוכנת `scp`

השימוש בתוכנה דומה מאוד לשימוש בתוכנת `cp` (העתקה) היוניקסית. ההבדל הוא שהיא מאפשרת להעתיק לכל שרת עם SSH – יש פשוט לכתוב את כתובת השרת ונקודותיים. אם מופיע נתיב לאחר הנקודותיים, הוא יחסי לתיקיית הבית. דוגמאות:

```
$ scp ex1.c USERNAME@t2.technion.ac.il:
$ scp USERNAME@t2.technion.ac.il:matam/ex1.c ./
```

2.3.2 תוכנת `sftp`

בקורס לימדו בדיוק איך להשתמש בתוכנת `ftp` הלא־מאובטחת. הגירסה המאובטחת, שמשתמשת ב־SSH (באופן זהה ל־`scp`), היא `sftp`. הפקודות של `sftp` זהות לפקודות של `ftp`, כפי שהוסברו בתרגול.

```
$ sftp USERNAME@t2.technion.ac.il
```

[5] הכוונה להפצות מודרניות ולמחשבי הלינוקס בחווה. כמובן שאם תתאמצו תוכלו בכיף להתארגן על הפצה שלוקח שעות על גבי שעות להתקין עבורה שרת X.

[6] זוהי ברירת מחדל שאפשר לשנות באמצעות פרמטרים בשורת הפקודה או קובץ בשם `Xresources`.

2.4 שרתי יוניקס אחרים

שרתי יוניקס אחרים שניתן להשתמש בהם בטכניון הם csl1 (לכל תלמידי הפקולטה), ו־tx (לכל בעלי הפרוטקציה). ראוי לציין את stud1 המנוח, יהי זכרו ברוך. ניתן גם להשתמש בכל מחשבי הלינוקס בחווה בתור שרתים - אך בבקשה אל תשתמשו במחשב לינוקס בחווה שמישהו אחר משתמש בו באותו רגע. לא בגלל שזה לא יעבוד (כי זה כן) – אלא בגלל שמדובר בפנטיום 3־ים עתיקים עם הארד־דיסקים גוססים. השימוש בכל אחד ואחד מהשרתים שצויינו כאן זהה לשיטות שמתוארות בשאר המסמך, למעת stud1 שהשימוש בו עשוי להצריך העלאה באוב.

2.5 אלטרנטיבות – עבור יוניקסופובים

אז החלטתם להמנע מכל מגע אפשרי עם יוניקס... ובכל זאת אתם רוצים שהסביבה שלכם יתאם את T2, בלי שתראה או תרגיש כמוה. למזלכם זה אפשרי ואף ישים. זה לא מומלץ כמובן... אבל הרבה יותר טוב מלעבוד עם הכלים של Microsoft Visual Studio.

סביבת העבודה שלכם תהיה dev-c++[7]. זוהי סביבת עבודה גרפית ונוחה, המכילה דיבאגר. כמובן שתצטרכו קומפיילר, ועדיף gcc. ישנה גירסא טובה של gcc לחלונות הנקראת mingw[8]. את רוב הסביבה הדרושה ניתן להשיג מ־Msys אשר נמצא באתר של mingw.

בנוסף תצטרכו את csh. כאן יש בעיה: אין csh לחלונות (שידוע לי עליו). פתרונות אפשריים:

- להעתיק ממישהו שהתקין Cygwin את csh.exe ואת cygwin1.dll (הדרך הפשוטה ביותר)
- לקמפל בעצמך את csh בעזרת הכלים לצוינו לעיל (אם הצלחת, תשלח לכולם שגם להם יהיה)
- להריץ את csh על מחשב אחר דרך ssh

בקיצור, עדיף להריץ Cygwin בשביל csh. אך כאן יש לשים לב מאוד: יש הבדל גדול בין הרצת סקריפטים של csh על T2 או על כל מחשב אחר. למשל, הסינטקס של less שונה, וגם דרושים כמה האקים כדי לגרום ל־T2 להיות יותר דטרמיניסטי בעת שימוש ב־csh.

3 לינוקס

3.1 מה זה לעזאזל לינוקס?

לינוקס הוא גרעין מערכת הפעלה שנכתב לראשונה ב־1990 ע"י סטודנט מדמ"ח פיני בשם לינוס טורוואלדס (שננשך מתישהו בשנת 2004 ע"י פינגווין). לינוקס הוא חסר־תועלת לחלוטין לבדו – בערך כמו kernel32.dll. דרושים כלי GNU כגון המעטפת, ls, cd, שרת X, ועוד. קבוצות רבות החליטו ליצור אוספים של כלים, שבמרכזם גרעין לינוקס, שמהווים יחד מערכת הפעלה. אוסף שכזה נקרא *הפצת לינוקס*. בחוות המחשבים בטאוב מותקנת הפצה בשם Ubuntu, וזוהי באופן כללי המלצת צוות Linux-Sup.

[7] http://www.bloodshed.net/devcpp.html
[8] http://www.mingw.org

3.2 מת"מ ולינוקס

ניתן להשתמש בלינוקס כדי להתחבר ל־T2 באמצעות SSH, כולל הפעלה גרפית של תוכנות – בשיטה שתוארה בסעיף "שרת X?!". אך יתרון גדול של לינוקס הוא שהשימוש בו הוא כמעט זהה לשימוש ב־T2, אך מקומי (כלומר, מהיר) ויותר נוח. כל הפצת לינוקס כוללת את כל התוכנות הרלוונטיות למת"מ – gcc, ddd, csh וכו', ולאחר שמתקינים אותם ממנהל החבילות של ההפצה, ניתן פשוט להפעיל אותם משורת הפקודה.

3.3 שימוש כללי בלינוקס

ניתן להשתמש בלינוקס לצרכים נוספים מלבד מת"מ. על כל מחשבי הלינוקס בחווה מותקן הדפדפן Mozilla Firefox, עם תוסף הג'אווה (עבור Mathnet למשל). מותקנות גם חבילות OpenOffice ו־LyX/LaTeX לצרכי עיבוד תמלילים, גליונות עבודה, מצגות וכו'. ניתן גם לראות הרצאות בווידאו באמצעות תוכנת vlc. הערה אחת שברצוננו להביא כאן היא לגבי הדפסה: נכון להיום, מנגנון ההדפסה מוגדר בחוות המחשבים בטאוב בצורה לא־נוחה במיוחד. על מנת להדפיס מסמך יש ליצור ראשית קובץ Postscript (אפשרי מכל תוכנה בלינוקס, בד"כ האפשרות מסומנת "Print to file"), ולאחר מכן להריץ את הפקודה הבאה:

```
$ askpcp filename.ps
```

יש לציין שבהתקנת לינוקס ביתית אין צורך בדבר שכזה, והגדרת המדפסת בהפצות מודרניות פשוטה מאוד – ומרגע שהיא נעשית, הדפסה אפשרית מכל תוכנה באמצעות מנגנון ההדפסה שלה.

3.4 הכלי make וקבצי Makefile

הכלי make מאפשר לבצע במהירות פעולות על קבצים בהתבסס על תאריך השינוי האחרון שלהם. השימוש המקורי והנפוץ ביותר שלו – קומפילציה של פרוייקטים (אך אפשר לעשות איתו הרבה יותר). כדי להשתמש בו, יש ליצור ראשית קובץ בשם Makefile (בלי סיומת, ורצוי אך לא חובה עם M גדולה). לא נפרט כאן יותר מדי על האפשרויות, אלא ניתן דוגמא שימושית:

```
CC=gcc
CFLAGS=-g -Wall -pedantic-errors -ansi
all: module1.o module2.o my_program
my_program:my_program.c module1.o module2.o
module1.o:module1.c module1.h
module2.o:module2.c module2.h
clean:
rm -f module1.o module2.o my_program
```

הקובץ הזה מכיל הוראות לגבי תלות וקומפילציה. השורה הראשונה קובעת שהקומפיילר שנשתמש בו לשפת C יהיה gcc (זו אינה ברירת המחדל על T2), והשורה השנייה קובעת אילו אפשרויות נרצה להשתמש בהן בכל קומפליציה. כל שורה עם נקודותיים מכילה מצד שמאל קובץ שאנו רוצים ליצור, ומצד ימין את הקבצים בהם הוא תלוי. מאחר ו־make יודע איך לקמפל קוד C, הוא ידע למשל שאם הסיומת של הקובץ שברצוננו ליצור היא o., אז יש להוסיף את האפשרות -c לקומפיילר. make גם יבדוק אם יש צורך בכלל לקמפל מחדש, הרי שייתכן שחלק מהקבצים לא השתנו ואין צורך לקמפל אותם מחדש.

יש לשים לב לשתי מטרות מיוחדות שהגדרנו – all ו־clean. שתיהן מטרות דמה, שהרי אין ברצוננו ליצור באמת קובץ בשם all או קובץ בשם clean. הסיבה ששמנו את all היא שביהעדר אפשרויות נוספות, make יבצע רק את המטרה הראשונה – אז יצרנו מטרה ראשונה מלאכותית שמקמפלת את כל הפרוייקט. המטרה clean, לעומת זאת, עושה ניקיון (שימו לב שהפקודה rm

מופיעה בשורה הבאה, עם ריווח בהתחלה), כדי שנוכל לקמפל מחדש באופן נקי או סתם כדי לחסוך מקום.

דוגמאות להרצת make, מהרגע שיש לנו את ה־Makefile מוכן:

```
$ make
$ make module1.o
$ make clean
```

השורה הראשונה תבצע את המטרה all, כלומר תקמפל את כל הפרוייקט. השורה השנייה תבדוק אם צריך לקמפל את module1.o, ותגיע למסקנה שלא בגלל שקבצי המקור שלא לא השתנו מאז שקימפלנו אותו (הרגע!). השורה השלישית תמחוק את כל הקבצים שקימפלנו, ותשאיר רק את קבצי המקור.

4 עורכי טקסט

עורכי טקסט זו כותרת גדולה?! כן. מאחר ויוניקס היא מערכת הפעלה המשמשת בעיקר מתכנתים, נושא עורכי הטקסט מאוד מפותח. ישנם עורכי טקסט רבים ומגוונים. בקורס הראו מעט אודות שימוש ב־pico וב־Emacs. ישנם עורכי טקסט פשוטים יותר לשימוש (טוב, לא הרבה יותר פשוטים מ־pico, אבל בכל זאת), כגון gedit ו־kate, שמותקנים על מחשבי הלינוקס. (שוב, כדי להריץ אותם – פשוט לכתוב את השם שלהם, ואם יש צורך אז את שם הקובץ אחריהם). עורכי הטקסט ה"גדולים" ביותר הם Emacs ו־vi ־ למעשה, משתמשי היוניקס מחולקים למחנות בין שני אלו. רוב סגל הפקולטה שייך למחנה ה־Emacs, ולכן קל למצוא מידע על העורך הזה (גם באתר הקורס). אנחנו דווקא ממחנה ה־vi, ולכן נביא כאן מידע על vi.

ראוי לציין שלכל עורך מכובד שהוא יש שלושה מאפיינים חשובים שיש להכיר: שימוש כללי (איך עובדים איתו), אופציות (מה תכננו שהוא יוכל לעשות), ואפשרויות להרחבה (מה לא תכננו שהוא יוכל לעשות, אבל הוא עושה בכל זאת). את הראשון צריך ללמוד ולזכור בע"פ, אבל את השניים הבאים בדרך־הכלל מאחסנים בקובץ הגדרות. ניתן למצוא קובץ הגדרות מוכן ל־Emacs באתר הקורס, וקובץ הגדרות מוכן ל־vi באתר של לוצקי: http://t2.technion.ac.il/~slutzky

4.1 מילה לגבי Emacs

עורך הטקסט Emacs הוא כלי מתוחכם ורב־עוצמה מבית היוצר של GNU. זה אינו עורך הטקסט האהוב עלינו, ולכן אין לנו הרבה ניסיון איתו. עם זאת, הוא קל מאוד לשימוש, ועדיין מתוחכם בהרבה מרוב עורכי הטקסט שראינו שנעשה בהם שימוש – pico, gedit, kate... ניתן כאן טיפ אחד שימושי מאוד – קומפילציה מתוך העורך:

1. צרו Makefile ע"פ ההוראות בהמשך מסמך זה.
2. בחרו בתפריט Tools→Compile. תופיע למטה שורת פקודה שתאותחל ל־make (ניתן להוסיף או להסיר פרמטרים כרצונכם). הקישו אנטר.
3. תורץ תוכנת ה־make, כלומר יתבצע ניסיון לקמפל את התוכנה.
4. החלון יפוצל, ובחלק שיפתח תופיע רשימת שגיאות הקומפילציה.
5. כדי לעבור לקובץ ושורה ברשימת שגיאת הקומפילציה, הביאו את הסמן לשגיאה הרצויה והקישו אנטר. העורך יעבור אוטומטית לקובץ והשורה של השגיאה.

באופן דומה, ישנם קישורים לתוכנות שימושיות אחרות מתוך Emacs, כגון gdb, כמו גם שפע משחקים קטנים לבזבוז הזמן.[9]

4.2 עורך הטקסט VIM

> בראשית בראו AT&T את sh ואת ed. ויוניקס היה תוהו ובוהו וסימן prompt
> על־פני רקע שחור, ורוח תביעות משפטיות עתידיות מרחפת על־פני המים.

התוכנה ed הייתה עורך הטקסט המקורי והקשה להחריד לשימוש של unix. בהמשך הגיע קונצפט חדשני לעולם – עורך טקסט שבעת עריכת הקובץ רואים את הקובץ שעורכים. עורך טקסט שכזה, אם אי פעם יצליחו לכתוב אחד, יקרא עורך טקסט ויזואלי. הצליחו, אז קראו לו vi. ואז שיפרו אותו, וקראו לו Vi IMproved, או בקיצור VIM. עורכי טקסט באו ועורכי טקסט הלכו, אך עד היום VIM הוא עורך הטקסט הותיק ביותר שעדיין בשימוש נפוץ ובפיתוח פעיל. הוא מותקן (אם כי בגירסה ישנה משהו) על T2, כולל את הגירסה הגרפית שלו – GVim. כמובן, כדי להפעיל אותם כותבים בפשטות vim או gvim (השני בהנחה שיש שרת X פעיל), עם או בלי שם־קובץ לעריכה. כאן יופיע הסבר קצר על השימוש בו, אך סביר להניח שאם מותקן על המחשב VIM, ניתן פשוט להקליד vimtutor ולקבל שיעור כבן 30 דקות בנושא.[10]

אני יודע מה אתם חושבים – מסובך, אני חוזר ל־pico. תהנו אני אומר. אבל באמת שאם תשקיעו את חצי־השעה הדרושה, סביר להניח שתמצאו ש־VIM הוא עורך טקסט מהיר לשימוש ונוח. נקודה לעידוד היא ש־VIM הוא ללא ספק אחת מהתוכנות המסובכות ביותר לשימוש ביוניקס, ואם אתם מצליחים להשתלט עליו (ובאמת שזה לא עד כדי כך מסובך), אתם יכולים להיות מאוד גאים בעצמכם.

4.2.1 מצבים

עורך הטקסט VIM הוא עורך מרובה־מצבים. כשמפעילים את העורך, המצב אליו נכנסים הוא מצב הפקודות, בו כל מה שמקלידים נחשב לפקודה. מצבים נוספים הם מצב ההכנסה (insert), בו VIM פועל כמו עורכי טקסט מודרניים אחרים, מצב ההחלפה (replace), מצב הסימון (visual), ומצב הסימון המלבני (blockwise visual). ישנם מצבים נוספים, אך לא נפרט עליהם כאן. הכניסה לכל מצב היא פקודה, היציאה מכמעט כל המצבים למצב הפקודות מתבצעת ע"י הקשה על Esc. דרך מעט נוחה יותר לחזור למצב הפקודות היא c^.[11]

4.2.2 שימוש בסיסי

ניתן כאן הסבר מאוד מאוד בסיסי על השימוש ב־VIM. מאוד מאוד מומלץ לעיין במסמכי :help.

כדי להתחיל להקליד יש להיכנס למצב insert. כדי לעשות זאת, יש להקיש i. במצב זה העורך פועל בצורה כמעט זהה ל־pico. כאשר מסיימים להקליד, חוזרים למצב הפקודות (Esc), ומקישים את הפקודות הבאות (עם הנקודותיים בהתחלה): :w כדי לשמור, ואז :q כדי לצאת. כדי לעשות את שניהם במכה: :wq.

ישנן אפשרויות רבות נוספות ל־VIM, כגון יישור אוטומטי, צביעת טקסט, השלמת שמות קבצים ועוד. את כל האינפורמציה ניתן להשיג ע"י הקשת הפקודה הבאה במצב הפקודות: :help, או :help *<topic>* עבור נושא ספציפי.

---

[9] בועז, בעת מחקר עבור המדריך: "איך אנשים מצליחים לתכנת בזה כשיש כאן טטריס???"

[10] נכון לעכשיו, vimtutor אינו מותקן על T2, אך כן מותקן על csl1.

[11] ניתן לראות כאן דוגמה לכך שב־VIM לא צריך להזיז את הידיים רחוק, ובדרך הכלל נשארים ב־"שורת הבסיס".

כדי להעתיק קטע מסויים, יש להיכנס למצב visual ע"י לחיצה על v ממצב הפקודות, לסמן את הקטע הרצוי, ולהקיש y. קיצור דרך: כדי להעתיק שורות שלמות, יש להקיש במצב הפקודות yy[*n*], כאשר n הוא מספר השורות להעתיק, ואם הוא לא יצויין אז תועתק שורה אחת. באופן זהה, הפקודה d גוזרת שורות והפקודה x גוזרת אותיות. כדי להדביק, יש להשתמש בפקודה (במצב הפקודות) p.

כדי לבצע חיפוש והחלפה, השיטה הכללית ביותר היא הפקודה הבאה: :%s/from/to/g. הפקודה תחליף את כל מופעי המילה from במילה to. סימן ה־% בתחילת הפקודה, משמעותו "בכל המסמך". השמטתו תגרום להחלפה רק בשורה הנוכחית. ניתן גם להחליפו בטווח מסויים. הפקודה s: תומכת גם בביטויים רגולריים (מידע בהמשך).

בעת עריכה של קוד מקור בשפת C או שפות דומות, שתי הפקודות הבאות שימושיות:

- :syntax on – הפעלת צביעת קוד אוטומטית
- :set cindent – הפעלת אינדנטציה אוטומטית

4.2.3 קומפילציה

באופן דומה ל־Emacs, גם ל־VIM יש מנגנון מובנה שמסייע עם make. מתוך מצב הפקודות, הקישו :make [*target*] (ניתן להוסיף פרמטרים). תורץ תוכנת ה־make, והעורך יעבור אוטומטית לשגיאה הראשונה. פקודות שימושיות לניתוח השגיאות:

- :cc – לתצוגת השגיאה הנוכחית
- :cnext – למעבר לשגיאה הבאה (בקיצור :cn)
- :cprev – למעבר לשגיאה הקודמת (בקיצור :cp)
- :clist – לרשימת השגיאות (בקיצור :cl)

4.3 עורך הטקסט joe

אם בכל זאת אתם רוצים עורך טקסט סביר, אבל Emacs ו־VIM מסובכים בשבילכם – אל דאגה, יש עוד עורכי טקסט (למעשה אלפים). joe הוא דוגמא לאחד פשוט וקל. אין לו את היכולות של VIM אבל קל ומהיר ללמוד להשתמש בו, והוא הרבה יותר שמיש מpico. (חוץ מהעובדה שDel לא עובד ב־T2)

כדי להכניס טקסט פשוט מקלידים אותו. כדי לשמור קובץ לוחצים ^K-D (^ זה המקש Ctrl). כדי לפתוח קבצים ^K-E (עם השלמת Tab, ראה tcsh), כדי לשמור ולצאת ^K-X. יש גם גזירה, הדבקה, העברה, undo, ועוד כמה פקודות, אותם ניתן נראות בחלון עזרה שמופיע כשלוחצים ^K-H

5 טיפים וטריקים כדי להקל על החיים

5.1 המעטפת tcsh

בקורס מלמדים את השימוש במעטפת csh. למעשה אתם תשתמשו במעטפת tcsh התואמת לה. יש שני דברים שימושיים שכדאי לדעת על המעטפת, שיחסכו לכם כתיבה.

הראשון הוא השלמת Tab. כאשר אתם מקלידים שם של קובץ, וכבר הקלדתם מספיק כדי שיהיה "ברור" על איזה קובץ אתם מדברים, לחצו על כפתור ה־Tab, ואם יש רק קובץ אחד עם

ההתחלה שכבר הקלדתם, המעטפת תשלים לבד את שם הקובץ. ההשלמה "חכמה", כלומר יודעת אם אתם רוצים תיקיה, קובץ הרצה או סתם קובץ כלשהו ולהשלים בהתאם.

השני הוא השימוש בכפתורים למעלה ולמטה. לחיצה על הכפתור למעלה תשים בשורת הפקודה את הפקודה האחרונה שנתתם לה. עוד לחיצה תיתן את הפקודה הקודמת לה, ואפשר להמשיך ללחוץ למעלה כדי לקבל פקודות קודמות. הכפתור למטה יתן את הפקודה שהכנסתם אחרי הפקודה שהוא מראה לכם.

5.2 המעטפת bash

זוהי המעטפת הסטנדרטית בכל הפצות היוניקס המודרניות. השימוש הבסיסי זהה ל־csh, ההבדל מתחיל בשימוש במשתני סביבה ובלולאות. עניין השלמת ה־Tab תקף גם כאן. כדי לעבור ל־tcsh, בהנחה שהוא מותקן, יש פשוט להקיש tcsh. עניין שראוי לתשומת לב הוא העברת פלט stderr לקובץ - הוא נעשה כאן בצורה יותר פשוטה מאשר tcsh. stderr הוא פשוט זרם מספר 2, וכדי להעביר אותו לקובץ, מבצעים פקודה בצורה כזו:

```
$ someprogram > OUTPUT.stdout 2> OUTPUT.stderr
```

ניתן לראות כי בדוגמה הזו זרם הפלט הסטנדרטי ישלח לקובץ OUTPUT.stdout, בעוד שזרם פלט השגיאות ישלח לקובץ OUTPUT.stderr. ניתן גם להעביר למשל רק את זרם השגיאות לקובץ, או רק את זרם הפלט הסטנדרטי.

5.3 צינורות

זוכרים redirection? ביוניקס זהו כלי מאוד שימושי. לעיתים רוצים לעשות משהו קצת שונה – להעביר פלט של תוכנה אחת ישירות בתור קלט לתוכנה שנייה. הדבר מאוד פשוט – משתמשים בסימן צינור (שיפט+\). לדוגמה:

```
$ program1 | program2
```

בדוגמה כאן, תרוץ ראשית התוכנה program1, עם קלט מ־stdin. הפלט שלה לא יופיע על המסך, אלא יועבר בתור קלט ל־program2. הפלט של program2 יופיע על המסך (אלא אם נוסיף צינורות נוספים או redirection אחר).

5.4 הכלי grep

זהו כלי יוניקסי שימושי לחיפוש בקבצים, שמותקן בכל מערכת יוניקס. השימוש בו פשוט מאוד:

```
$ grep 'SEARCH STRING' [files]
```

הפלט יהיה כל השורות מהקבצים המכילות את המחרוזת SEARCH STRING (נא לשים לב לגרשיים – יש בהם צורך במקרה שיש רווחים במחרוזת החיפוש). אם לא יצויינו קבצים, אז החיפוש יערך על זרם הקלט הסטנדרטי. למשל, כדי להראות את כל השורות עם המילה TODO בקובץ מסויים, ניתן לבצע אחת משתי הפקודות הללו:

```
$ grep TODO somefile.c
--OR--
cat somefile.c | grep TODO
```

5.5 הכלי sed

זהו האולר השוויצרי של עיבוד טקסט ביוניקס. השם הוא קיצור ל־Stream EDitor, וזה בדיוק מה שהתוכנה עושה – היא מבצעת עריכה מסויימת על זרם הקלט שלה. התוכנה מאוד מתוחכמת ויש לה מגוון אדיר של אפשרויות, אבל השימוש הבסיסי מתבצע כך:

```
$ program1 | sed 's/From/To/'
```

התוכנה program1 תופעל, והפלט שלה יועבר ל־sed. sed תפלוט את הקלט שלה, אך תחליף בכל שורה את המילה From למילה To. במקרה זה הפלט יופיע על המסך, אך אפשר כמובן להעביר אותו לקובץ או לתוכנה אחרת. שימו לב שכאן המילה תוחלף רק פעם אחת בכל שורה. כדי להחליף את כל ההופעות של From במילה To, הפקודה היא:

```
$ program1 | sed 's/From/To/g'
```

ניתן להשתמש במחרוזת החיפוש גם בביטויים רגולריים (ראו הסבר בהמשך). בזכותם sed הוא כלי רב עוצמה במיוחד. כמה רב עוצמה? ובכן, מספיק לאמר כי *אסור בתכלית האיסור להשתמש ב־sed בתרגיל ב־csh*.[12][13]

5.6 הכלי valgrind

זהו כלי מתוחכם המשמש בעיקר לאיתור דליפות זכרון בתוכנה. כלי זה זמין כיום רק ללינוקס על מחשבי x86 – כלומר לא לחלונות, לא ל־T2, אלא רק לתחנות הלינוקס (שמריצות אובונטו) או מחשבי לינוקס על x86 שיש לכם בבית. השימוש ב־valgrind נעשה כך:

```
$ valgrind my_program
```

כאן תורץ התוכנה my_program (שצריכה להיות בתיקייה הנוכחית, ורצוי שתהיה מקומפלת עם g-), ובסיום ההרצה ידווחו שגיאות גישת זכרון ודליפות, יחד עם הוראות להוצאת מידע ספציפי יותר מ־valgrind. השגיאות הנפוצות ביותר ש־valgrind מזהה הן אי־שחרור זכרון שהוקצה (valgrind סופר בדיוק כמה בתים הוקצו וכמה שוחררו) וגישה לזכרון (משתנים) שלא אותחל.

5.7 ביטויים רגולרים (regexps)

ב"רים הם אחד הכלים השימושיים, חזקים ונוחים שנפלו לידיהם של ציבור המתכנתים (או במילים אחרות, לא מלמדים את זה במת"מ כי זה עושה חיים קלים מדי). זהו כלי לחיפוש והתאמה והחלפה של טקסט. הוא הועבר לכמעט כל שפת תכנות מודרנית, קיים בכל מערכת הפעלה, במסרי נתונים רבים, בכמעט כל עורך טקסט, ושימוש בהם יכול לחסוך לכם הרבה מאוד טרחה.

5.7.1 תחביר בסיסי

היות וזה מדריך כללי ולא מדריך ב"ר, יהיה פה רק חומר בסיסי ביותר. לצרכי הדגמה נשתמש בפסודו־תחביר הבא להפעלת ב"ר על משתנה

```
var ~= /a/;
// True if pattern a is found in var (match)
```

נתחיל עם הפעולה הבסיסית ביותר: חיפוש מחוזת במחרוזת. אם אנחנו רוצים למצוא את המחרוזת "foo" במשתנה bar , נפעיל את הביטוי

```
bar ~= /foo/;
```

פשוט, לא? פשוט הצבנו את "foo" במקום a. אם המשתנה bar מכיל את המחרוזות "aaafooaaa" או "a foo adaa" יוחזר אמת, אבל "f o o" יחזיר שקר.

נעבור למשהוא יותר שימושי: תווים לא ידועים. הם מסומנים ב '.', כלומר אם אני מחפש מחרוזת שיש בה 'h' ואז תו כלשהוא ואז 't', במשתנה bar נכתוב:

```
bar ~= /h.t/
```

[12]...כי אז היה לוקח בערך רבע שעה לעשות אותו...

[13]...וזה אם בחיים לא השתמשתם ב־sed.

ואז נקבל אמת בשביל "hat", "h@t", "htt" אבל לא בשביל "ht" או "haat".
מה אם רוצים מספר לא ידוע של תוים? בשביל זה יש את הסימן '*'. הסימן מתיחס לתו לפניו. משמעותו – התו הזה, 0 או יותר פעמים. כלומר:

```
bar ~= /bo*m/
```

יחזיר אמת למחרוזות "bm", "bom", "boom", "booom" וכו'. אם רוצים אחד או יותר משתמשים בסימן '+'. עוד כמה סימונים:

- סוף מחרוזת '$'
- תחילת מחרוזת '^'
- קיבוץ טקסט '(',')' . (כלומר סימונים כמו '*' יפעלו על כל הטקסט בין הסוגריים במקום תו אחד)
- טווח אורכים '{a,b}'. (מחליף את '*'. משמעותו a עד b פעמים. אם אין b אז a עד אינסוף)
- אפס או אחד פעמים '?'.
- טווח תוים '[',']', (מחליף את '.', משמעותו אחד מן התוים בסוגריים)

ניתן להשתמש בב"רים גם להחלפה. הפקודה s משמעותה swap (החלפה), כמו ב־sed:

```
var ~= s/a/b/
```

נאמר שאנחנו רוצים שבין שני a־ים יבוא b. נבצע את ההחלפה:

```
var ~= s/a.*a/aba/
```

הביטוי הזה יהפוך את "another take" ל "abake", את "aa" ל "aba", ואת "tattattat" ל "tabat". שימו לב שבדוגמא השלישית הוא לקח את ה־a הרחוק ביותר. זה נקרא פעולה חמדנית. אם מוסיפים את הסימן '?' אחרי ה־'*' תהפוך את פעולה למינימלית במקום חמדנית, כלומר הפעולה

```
var ~= s/a.*?a/aba/
```

תהפוך את "tattattat" ל"tababat". (בהנחה שאמרתם להחליף הכול ולא רק מקרה אחד, ראו שימושים)
אפשר לבקש מההחלפה לזכור דברים. זה נעשה על ידי הכנסתם לסוגריים. את הדבר הראשון בסוגריים ההחלפה תזכור כ־$1, את השניה כ $2 וכו. אפשר להשתמש הסימונים במחרות תחליף (כלומר מחרוזת b). לדוגמא אם אנחנו רוצים להפוך את "<x,y>" ל "(x,y)" כאשר x,y ספרות בין אחד ל־3, נפעיל את הביטוי

```
var ~= /<([123]),([123])>/\($1,$2\)/
```

שוב שימו לב לשימוש ב־'\' כדי לציין שלתו אין משמעות מיוחדת.
זה מספיק ידע לשימושים רבים ומגוונים. למדריך שימושי לב"רים מומלץ לפנות לאחד מהאתרים הבאים:

- http://analyser.oli.tudelft.nl/regex/index.html.en
- http://en.wikipedia.org/wiki/Regular_expression

5.7.2 שימושים

ישנם מאות שימושים. נציין רק כמה מהם. נתחיל מהפקודה grep. זו פקודה של שורת הפקודה של unix. היא מקבלת ב"ר וקובץ, קוראת את הקובץ ומדפיסה רק את השורות שמכילות את הב"ר. אם לא ניתן לה קובץ היא קוראת מהstdin (וזה מתי שהיא באמת שימושית). התחביר שלה הוא:

```
> grep regex [file]
```

דוגמא לשימוש הוא להדפיס את שמם של כל הקבצים שהסיומת שלהם היא c. או h. בתיקיה הנוכחית:

```
> ls -l | grep "\.[ch]$"
```

עכשיו שיש לנו רשימה, עוד שרשורים לפרודה יאפשרו לנו לבצע פעולות על כל הקבצים ברשימה (לדוגמא להוסיף אותם לMakefile). שימו לב שה'\' משמעותו שהנקודה היא נקודה רגילה ולא תו כלשהוא. בשביל עוד מידע אפשר לכתוב בשורת הפקודה

```
> man grep
```

כדי להקל על השימוש ב־man, ניתן גם שם להשתמש בב"רים. כי לחפש לוחצים על המקש / ומקלידים את הביטוי. לחיצה על Enter תיקח אתכם להתאמה הראשונה. לחיצה על 'n' תיקח אתכם להתאמה הבאה. שימוש זה גם נכון ל־vi במצב פקודה. כדי להחליף ב־vi משתמשים ב:

```
:%s/a/b
```

כאשר a הוא הביטוי, ו־b הוא התחליף. יש גם לשים לב שכדי להגיד לביטויים קודמים ב־vi משתמשים ב־\1 במקום $1.

בEmacs ניתן ללכת לתפריט הבא:

Edit → Search → Advanced Search & Replace → Regex Search

וכמובן יש Regex Replace.

ישנן ספריות ל־C, C++, C#, Java, VBasic וכו, שמאפשרות שימוש בב"רים, והם פקודות מובנות בשפות כמו Perl, Python ו־PHP. במיוחד Perl.

5.8 ללמוד שימוש בסיסי ב־perl

פרל היא שפת תיכנות שנוצרה במטרה לאפשר אוטומציה של דברים פשוטים[14] (כמו להתחבר לרשת של טאוב מלפטופ). היא נועדה להיות מוכרת ופשוטה למי שיודע C או shell. ידע בסיסי בה שימושי לחלונות באותה מידה שהוא שימושי ליוניקס. לא נביא פה מדריך בסיסי לפרל, אבל נמליץ על כמה:

- http://code.semuel.co.il/perlhebtut/ (עברית)
- http://perl.eitan.ac.il/main.php?id=00006 (עברית)
- http://www.comp.leeds.ac.uk/Perl/start.html
- http://archive.ncsa.uiuc.edu/General/Training/PerlIntro/

[14] קרי: תכנות לעצלנים עוד יותר