אוניברסיטת חיפה
החוג למדעי המחשב
מערכות הפעלה – תרגול

תרגול מס' 1

מתרגל: שי אררה

סלולרי: 052-8444747

כתובת אימייל: serera@gmail.com

משימות קורס:

3-4 משימות (יפורסמו באתר).

אחוז מהציון בקורס – 5%.

הגשה בזוגות.

מטלות נוספות:

- **פרויקט**: אחוז מהציון – לפי החלטת המרצה.
- **מבחן**.

חומר עזר:

- **Abraham Silberschatz & Peter Galvin - Operating System Concepts – חובה !!!**
- **רצוי לקרוא על**:
  - יצירת DLL ב-Visual Studio .NET
  - יצירה של Threads ב-C/C++

תרגולים:

יום **חמישי** 10:00-12:00 ויום **שישי** 08:30-10:00.

שעות קבלה: בתיאום מראש.

אוניברסיטת חיפה

החוג למדעי המחשב

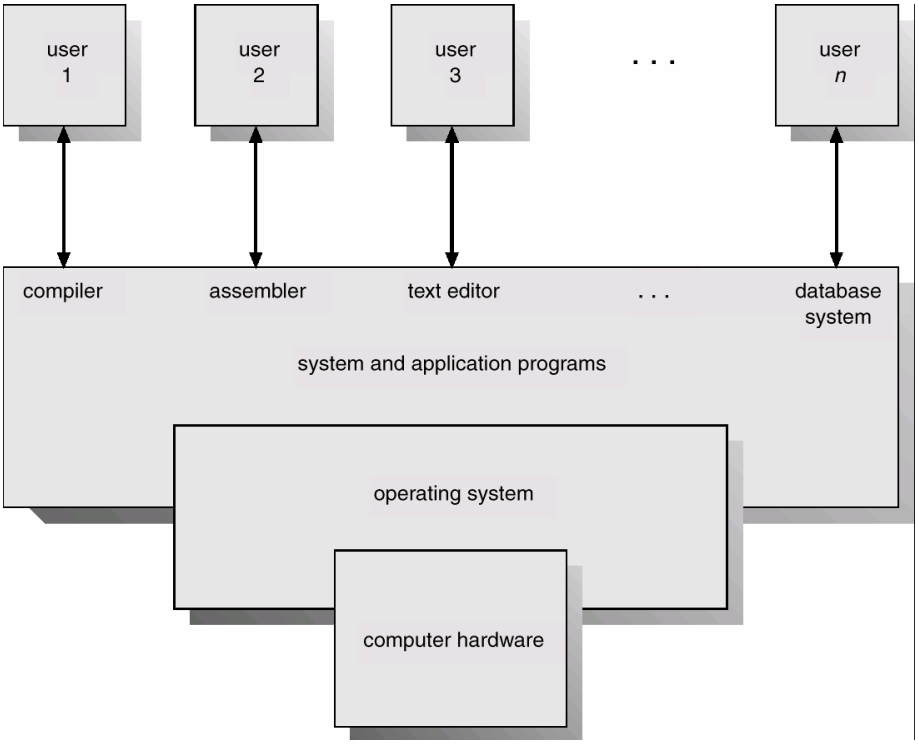
מערכות הפעלה – תרגול

נושא התרגול: הכרת מערכת ההפעלה Unix

מערכת ההפעלה היא המוח / הלב של המחשב. היא הגורם המקשר בין התוכניות השונות שרצות במחשב לבין החומרה של המחשב.

<u>תפקידים שונים</u>:

- מספקת למתכנתים סביבת High-Level לתקשורת עם החומרה במחשב.
- מספקת למשתמשים ממשק נוח לעבודה עם התוכניות השונות.
- מנהלת זיכרון (באופן יעיל? .... נראה בהמשך ☺).
- מנהלת את הגישות של המשתמשים להתקני החומרה השונים.
- ...



מערכת ההפעלה מספקת שירותים (Services) למשתמשים.

השירותים הללו מסופקים דרך תוכניות מערכת (System Programs) שונות, אשר משתמשות בקריאות מערכת (System Calls).

נשמע מסובך – אל תדאגו, זה יסתבך יותר ...

מהן למעשה System Calls?

ובכן, System Calls למעשה מספקות שירותים שונים למתכנתים ולמערכת ההפעלה. למשל:

- שליטה ב-Processes שרצים במערכת (סיום תהליך, התחלת תהליך וכו').

- ניהול מערכת הקבצים (File System).
- ניהול התקני החומרה השונים (בדרך כלל ע"י שימוש ב-Device Driver).
- תקשורת.
- ...

תוכניות המערכת (System Programs) הן למעשה אותן תוכניות שהמשתמש מריץ (במודע או שלא במודע).

נבחן מערכת הפעלה עלומת שם ונראה כיצד הדברים הללו באים לידי ביטוי.

מערכת ההפעלה Unix

ה-SHELL

בכל מערכת הפעלה, אנחנו עובדים בתוך Shell (קליפה) שהמערכת מספקת לנו. דרך ה-Shell אנחנו יכולים לתקשר עם מערכת ההפעלה, החומרה והזיכרון באמצעות ... System Programs.
כאשר מדליקים את המחשב, התוכנית הראשית של מערכת ההפעלה (Kernel) נטענת לזיכרון הראשי (במערכות של היום נטענות עוד הרבה תוכניות, וביניהן ה-Shell). ה-Kernel אחראי לביצוע פונקציות מרמה נמוכה ורמת System. בנוסף, הוא אחראי על זימון תהליכים וביצוע פעולות I/O.
ה-Shell היא תוכנית משתמש הנקראת גם "Command Interpreter". זוהי בד"כ התוכנית המופעלת עם כניסתו של המשתמש למערכת (login). מטרת ה-Shell היא להיות שלב ביניים בין ה-Kernel למשתמשים. ה-Shell מתרגם פקודות משתמש לפעולות המתבצעות ע"י ה-Kernel. בנוסף ה-Shell מאפשר למשתמש להריץ תוכניות, לחפש קבצים ועוד.

1. תהליך הכניסה למערכת (login)

כדי להיכנס למחשב יש לדווח לגרעין את זהות המשתמש ואופן התקשורת אתו. לשם כך הגרעין מפעיל לכל User Port (tty - terminal type) את התוכנית getty. התוכנית מציגה login prompt.
כאשר התכנית getty מקבלת קלט, נקראת התוכנית login. התוכנית הנ"ל מזהה את המשתמש ובודקת אם יש לו זכות להיכנס למערכת. אם ה-password של המשתמש נכון, מופעלת התוכנית שרשומה ב-password file. זו יכולה להיות כל תוכנית אך בדרך כלל זהו ה-Shell.

2. תפקידי ה-Shell

ה-Shell מיועד למשתמש בודד ומשמש את הצרכים שלו. במילים אחרות זהו ממשק בין המשתמש ומערכת ההפעלה. למשתמש יש אפשרות לכתוב Shell משלו.

התפקידים הבסיסיים של ה-Shell:

- Command Line Interpretation
- Program Initiation
- I/O Redirection
- Pipeline Connection
- Substitution of Filenames
- Maintenance of Variables
- Environment Control
- Shell Programming

2.1. Command Line Interpretation

כאשר המשתמש מבצע login מופעלת גרסה אינטראקטיבית של ה-Shell. המשתמש מכניס פקודות ל-Shell Prompt וה-Shell מנסה לפרש אותם. הפורמט הבסיסי של פקודה:

> command <arguments>

לדוגמה:

> ls -l file1 file2

ה-Shell מזהה את הפקודה ls ושלושה ארגומנטים.

ב- Shell קיימים שלושה סוגי פקודות:

1. קובץ לביצוע (Executable File) שנוצר ע"י קומפילציה של תוכנית הכתובה בקוד מסוים.
2. קובץ המכיל מספר פקודות Shell (Script - יפורט בהמשך).
3. פקודות פנימיות של ה- Shell הנקראות גם פקודות built-in.

2.1.1. Built-in Commands

חלק מהפקודות ממומשות ע"י ה-Shell עצמו. כלומר, פקודות אלה אינן גורמות ל-Shell להריץ תוכנית חיצונית, אלא לבצע קוד פנימי (פונקציות) של תוכנית ה-Shell.
ביצוע פקודות כאלה נעשה בצורה מהירה כי אין צורך להריץ שום תוכנית חיצונית. בטבלא הבאה מופיעות פקודות built-in עיקריות.

Command	Description
alias	Define or list a command alias
bg	Background execution
fg	Switch a job to foreground execution
cd	Change directory
history	List command history
jobs	List active jobs
kill	Kill a process
set	Display or change a variable
setenv	Set environment variable
unsetenv	Delete environment variables

הסבר על חלק מפקודות ה-built-in:

Alias

המשתמש יכול להגדיר שמות משלו לפקודות קיימות. למשל, נניח שאנחנו רגילים לעבוד ב-DOS ואנחנו מכירים את הפקודה dir ולא ls –l. אז אנחנו יכולים להקליד alias dir ls –l ובכל פעם שנקליד dir, ה-Shell יחליף את זה ב-ls –l.

History

ה-Shell שומר את ההיסטוריה של הפקודות שהוקשו ע"י המשתמש. אם נקיש history נראה את רשימת הפקודות שביצענו בסדר עולה.

2.2. Program Initiation

כאשר ה-Shell מתרגם את ה-Command Line הוא מזהה את הפקודה שיש לבצע. אם זאת לא פקודת built-in ה-Shell מחפש את קובץ ה-executable (בעזרת המשתנה PATH) וגורם לגרעין לבצע את התוכנית.

הרצת תהליכים ב- Shell:

לכל תהליך במערכת יש מספר אשר מזהה אותו. הרצת תהליך תוך המתנה לסיומו נקראת הרצה ב-**foreground.** אם אנו רוצים להריץ תהליך ובזמן שהוא רץ, להמשיך לעבוד, נריץ אותו ב-**background.** נעשה זאת ע"י הסימן &.

הפקודה הבאה גורמת להרצת **find** ב-**background** תוך הפניית הפלט ל-**res**, ובינתיים ניתן להמשיך לעבוד.

find /usr -name hello.c > res &

פקודות נוספות לשליטה בתהליכים הן.

1. **fg -** להעברת תהליך מ-**background** ל-**foreground**.

2. **bg -** להרצת תהליך ב-**background**

3. **kill -** להפסקת ריצתו של תהליך - <מספר תהליך> **kill**,

4. לחיצה על **ctrl-z** משהה את ביצועו של תהליך שרץ ב-**foreground** .

הפקודות **fg, bg** פועלות גם על תהליכים מושהים, ז״א ניתן להמשיך את ריצתו של תהליך שהושהה ב-**background** ע״י הקשת **bg**.

2.3. I/O Redirection

המשתמש יכול להגדיר לקליפה מהיכן לקחת את הקלט לתוכנית (ברירת מחדל-מקלדת) ולהיכן להוציא את הפלט של התוכנית (ברירת מחדל-מסך). לכל תוכנית 3 ערוצי קלט/פלט.

STDIN - Standard Input

STDOUT - Standard Output

STDERR - Standard Error

הפקודות הבאות מדגימות את פעולת ה- **redirection** :

myprog < input.txt > output.txt

(myprog < input.txt > output.txt) > & error.txt

2.4. Pipeline connection

למשתמש ניתנת האפשרות להעביר את הפלט הסטנדרטי של פקודה מסוימת כקלט סטנדרטי לפקודה הבאה אחריה. קישור זה נעשה ע״י הפרדת שתי הפעולות בסימן |. לדוגמא, אם ברצוננו לספור כמה מילים נותנת לנו הפקודה **ls** על המחיצה הנוכחית, נקשר בין הפקודה **ls** לבין הפקודה **wc** הסופרת מילים בצורה הבאה: **ls | wc**, הקליפה מפעילה את **ls** תוך קישור ה-**Standard Output** לקובץ **PIPE** מיוחד ובאותו זמן מופעל גם **wc** ע״י הקליפה, תוך קישור ה-**Standard Input** לאותו קובץ **PIPE**. כך נוצר מצב שהפלט של **ls** מהווה את הקלט של **wc**.

2.5. Substitution of Filenames

אפשר להשתמש בתווים מיוחדים כדי להתייחס ליותר מקובץ אחד. ה-Shell מבצע את ההחלפה לפני ביצוע תוכנית. לדוגמה:

```
> echo *
file1 file2 file3 file3x file4
```

כדי למנוע מה-Shell לפרש את הסימנים יש להשתמש בגרשיים.

```
> echo "*"
```

2.6. Maintenance of variables

המשתמש יכול להגדיר ל-Shell משתנים כרצונו. לדוגמא, אם ברצוננו להגדיר את המשתנה COMPILER נבצע:

> set COMPILER=gcc

כאשר נרצה להתייחס למשתנה שהגדרנו נציב לפניו את סימן ה $ לדוגמא:

> set name=Gal

> echo hello $name

> echo hello $(name)-San

בפקודה השנייה הוספנו סוגריים כי רצינו להצמיד את המשתנה לתו אחר.

2.7. Environment Control

הסביבה שהמשתמש עובד בה מכילה:

- Home directory
- Type of the terminal
- Path that will be searched for executable files

הסביבה נשמרת במשתני סביבה כגון PATH, TERM, HOME. פקודת setenv מגדירה משתנה סביבה חדש או מציבה ערך חדש במשתנה קיים. לדוגמה:

> setenv TERM vt100

ניתן להתייחס למשתני סביבה באותו אופן שמתייחסים למשתנים רגילים.

תוכנית שרצה תחת Shell יכולה לגשת למשתני סביבה של אותו Shell.

2.8. Shell programming

ראינו שה-Shell יודע לפרש command line, ולהריץ תוכניות. אולם ניתן גם לתכנת בסביבת Shell.

הדבר דומה לכתיבת קבצי Batch (BAT) ב-DOS. למעשה אנו כותבים מעין Script אשר מדמה הקלדה של פקודות באופן סידרתי ע"י המשתמש. בנוסף, ניתן להשתמש בפקודות מיוחדות לתכנות ב-Shell (כמו לולאות, תנאים וכו').

על מנת להריץ פקודות מתוך קובץ, יש להגדיר את מיקום תוכנית ה-Shell אשר תבצע את שורות הקובץ בשורה הראשונה בקובץ לדוגמא:

#!/bin/csh -f

בנוסף יש לעדכן הרשאות הקובץ (באמצעות הפקודה chmod) כך שניתן יהיה להריץ אותו.

אוניברסיטת חיפה

החוג למדעי המחשב

מערכות הפעלה – תרגול

סט הוראות שמיועדות בעיקר לתכנות ב-Shell.

1. לולאה שבה המשתנה מקבל כל ערך מערכי הרשימה var-list

```
foreach  var  <var-list>
        commands
        ......
end
```

ניתן גם להשתמש ב break ו continue כמו ב-C. ברשימה ניתן גם לכתוב ביטויים כמו הביטוי *.c

2. ביצוע מותנה: expr מסמל ביטוי בוליאני אשר מחזיר את הערך 0 (ייחשב כ"שקר") או ערך שונה מ- 0 (ייחשב כ"אמת").

```
if (expr) then
...
else
...
endif
```

ביצוע מותנה עפ"י switch:

```
switch (var)
        case  val1 :
        ...
        breaksw
        case  val2 :
        ...
        breaksw
default :
        ...
        breaksw
endsw
```

3. לולאת while:

```
while (expr)
...
end
```

הפרמטרים של קובץ script

כאשר אנו מבצעים קובץ המכיל פקודות, ניתן להתייחס אל הפרמטרים המועברים אליו בצורה הבאה:

1. $0 - שם קובץ הפקודות
2. $<מספר> - התייחסות לפרמטר (עפ"י המספר)
3. $* - שרשור של כל הפרמטרים

4. $# - מספר הפרמטרים שהועברו לקובץ

דוגמאות לקובצי ביצוע:

```
# Example A
# go over all directory C files and
# show their content
foreach var ( *.c)
echo "file is " $var "----------"
```

```
# Example B
# set a variable and print its parts
set var=(abc def  ghi  jkl)
foreach i (1 2 3 4)
echo "part number" $i "is:" $var[$i]
```

- "***.c**" מייצג את רשימת הקבצים בעלי סיומת **c** במחיצה הנוכחית
- ניתן להתייחס למילים בתוך משתנה (כמו התייחסות למערך).

אוניברסיטת חיפה

החוג למדעי המחשב

מערכות הפעלה – תרגול

פקודות שימושיות של Unix

פקודה	הסבר	דוגמה	הסבר	דוגמה נוספת
man	קבלת עזרה	**>man command**	נותנת הסבר על הפקודה command	**>man –s open**
ls	רשימת הקבצים	**>ls –l**	רשימה ארוכה	**>ls –al**
cat	הדפסת קובץ על המסך	**>cat temp.c**	הדפסת קובץ temp על המסך	**> cat *.c**
more	הדפסת קובץ מסך- מסך	**>more temp.c**		**>more *.h**
grep	**grep [options] word [files]** מחפשת בקבצים [files] או בקלט הסטנדרטי שורות בהן מופיעה המילה word ומוציאה אותם לפלט.	**>grep Thursday diary**	מדפיסה את שורות הקובץ diary בהן מופיעה המילה **Thursday**	**>grep printf *.c**
find	מציאת קובץ שאיננו יודעים את מקומו	**>find /usr –name diary**	מחפשים את הקובץ diary. תחילת החיפוש במדריך /usr	**>find $HOME -name koko**
ps	קבלת תהליכים רצים במערכת	**>ps –a**	רשימה של כל התהליכים במערכת	
sort	מיון שורות של קובץ	**>sort –r temp**	הדפסת שורות הקובץ temp בסדר לקסיקוגרפי הפוך	**>cat temp |sort -r**
which	מציאת מיקום הגדרת תוכנה. זוהי פקודת built in	**>which gcc**	מציאת מיקום המהדר gcc	**>which set**
tar	יצירת ארכיבים	**>tar cf dir.tar**	יצירת ארכיב בשם dir.tar	**>tar xf dir.tar *.***
gzip	דחיסת קובץ	**>gzip file.c**		**>gunzip file.c.gz**
wc	**wc [-c | -l | -w] [filename1 ....]**: ללא אופציות, מדפיסה wc את מספר השורות, מספר המלים ומספר האותיות בקלט שלה או בקבצים filename1, …	**>wc main.c .login**		**>cat file.c | wc**
head	**head [-n] [filename]** הפקודה מדפיסה n שורות (או 10) מראשית הקובץ filename או מהקלט שלה	**>head –5 jargon**	הדפסת 5 השורות הראשונות של הקובץ jargon	
tail	דומה ל head, אך מסוף הקובץ	**>tail –3 jargon**		
chmod	שינוי הרשאות של קובץ	**>chmod u+x test>**	מתן הרשאות ביצוע לקובץ test	**>chmod u-r test**
cd	מעבר למחיצה אחרת.	**>cd /usr/bin**		**>cd**