1. משתנים ב c-shell

1.1. משתני סביבה

מוגדרים ע"י הפקודה **setenv** , וניתן לבטל אותם ע"י **unsetenv**.

משתנה סביבה חייב להכיל מחרוזת אחת בלבד :

```
t2:eesoft>  setenv  MyEnv  nothing
t2:eesoft>  setenv  YourEnv  "nothing"
t2:eesoft>  setenv  OurEnv  "nothing meaningfull"
t2:eesoft>  unsetenv YourEnv
t2:eesoft>  setenv
MyEnv = nothing
OurEnv = nothing meaningfull
t2:eesoft>  echo  my environment variable equals $MyEnv
my environment variable equals nothing
```

הפקודה setenv ללא פרמטרים, מציגה למסך את כל משתני הסביבה שקיימים ב shell הנוכחי.

כל תוכנית שמורצת מה- c-shell מקבלת את רשימת משתני הסביבה שהיו ל shell בזמן ההרצה, וניתן לגשת אליהם ע"י פקודת ה-C getenv() :

```
/* env,c: an example for using getenv() */
#include <stdlib.h>       /* to use getenv( ) */
#include <stdio.h>

void main( ) {
        char*       val;

        val = getenv("MyEnv");

        if (val == NULL)
                printf("Env. variable not found\n");
        else
                printf("Env. variable contains %s",val);
}
```

1.2. משתני shell

להבדיל ממשתני סביבה, משתני shell :

- **לא עוברים** לתוכניות שמורצות מה c-shell.
- יכולים להחזיק **מערך של ערכים** ואין הגבלה של ערך בודד למשתנה.

מגדירים משתני shell ע"י **set** , ומבטלים ע"י **unset**.

כדי להגדיר מערך :

```
t2:eesoft>  set  x = ( a  b  c  d )
t2:eesoft>  echo $x   $#x
a  b  c   d  4
t2:eesoft>  echo $x[2]
b
t2:eesoft>  echo $x[1-3]  $x[2-]
a b c b c d
```

1.3. command substitution:

ניתן להציב למשתנה פלט של פקודה אחרת, דוגמא:

```
t2:eesoft>  set Z = ` cat file | wc –l `
t2:eesoft>  echo  $Z
14
t2:eesoft>
```

2. החלפת שמות קבצים

לתוים *,[ ],? ,~ משמעות מיוחדת בשורת פקודה ל shell, והם מוחלפים לרשימה ממוינת של שמות קבצים/מדריכים לפי הדוגמאות הבאות:

```
t2:eesoft>  ls –l  *.c
```

מראה את רשימת כל הקבצים המסתיימים ב- c.

```
t2:eesoft>  ls   a*
```

כל הקבצים שמתחילים ב- a.

```
t2:eesoft>  echo  [a-dfg]xy.[cC]
```

מציג את שמות כל הקבצים שהאות הראשונה בהם a,b,c,d,f,g, השניה x השלישית y, והסיומת היא c או C.

```
t2:eesoft>  cat  fileno?.txt
```

מציג למסך את תוכן כל הקבצים עם שם שמתחיל בשש האותיות fileno, בנוסף לעוד אות (שלא משנה מה היא) , ומסתיים ב txt..

```
t2:eesoft>  cd  ~/mail
```

חוזר לתת-מדריך mail של מדריך הבית של המשתמש.

```
t2:eesoft>  ls ~ilana/
```

מציג את רשימת הקבצים במדריך הבית של המשתמש בשם ilana.

3. בקרת זרימה

3.1 פקודות (ראו פירוט בהרצאה)

3.1.1 if

```
if ( condition ) simple-command
```

או

```
if (condition ) then
        command list
[else
        more command list]
endif
```

3.1.2 while

```
while (condition )
        command list
end
```

3.1.3 foreach

```
foreach  var ( wordlist )
        command list
end
```

3.1.4 switch

```
switch ( string )
        case pattern1:
                command list1
                breaksw
        case pattern2:
        ..
        default:
                command listX
endsw
```

3.2 אופרטורים ב c-shell

כפי שנוכחתם לראות בחזרה לעיל, הפקודות לבקרת זרימה מחייבות הגדרת תנאים (conditions) ותבניות (patterns), לצורך זה הוגדרו אופרטורים:

!	not
*,/,%,+,-	כפל, חיסור, חיבור, שארית, חלוקה
&&,||	logical or, logical and
>=,<=,<,>	גדול מ-, קטן מ-, קטן-שווה, גדול-שווה.
==,=!,=~,!~	שווה, לא-שווה, תבניות שוות, תבניות לא שוות

הסט האחרון של אופרטורים עובד על מחרוזות ומשמש לעבודה עם תבניות, כל השאר עובד עם מספרים.

3.3 ביטויים :

כל ביטוי ב- c-shell, מחזיר ערך 1 או 0 , ויכול לשמש כתנאי (condition) , עבור פקודות בקרת זרימה, דוגמא:

לבדוק אם מחרוזת מכילה תו נתון:

```
t2:eesoft>  if  ( “ this is a string” =~ *a*)  echo “has a”
has a
```

לבדוק אם מחרוזת מסתיימת באות z או Z:

```
t2:eesoft>  set  end_with_z = "amos oz"
t2:eesoft>  if (  “$end_with_z”  =~  *[zZ])  echo match
match

t2:eesoft>  set  no_end_with_z = clinton
t2:eesoft>  if (  $no_end_with_z =~ *[zZ])  echo match
t2:eesoft>
```

לבדוק, אם הקובץ הוא h או c , אך לא מתחיל במילה BST

```
t2:eesoft>  if ($filename =~ *.[CHch] && $filename !~ BST*)
……..
```

כמו כן, מומלץ מאוד להשתמש בתבניות בעבודה עם **switch**.

3.4. שאילתות על קבצים:

בנוסף לביטויים הרגילים, ניתן להגדיר תנאים לפקודות בקרת הזרימה בעזרת שאילתות על קבצים:

```
t2:eesoft>  if ( ! ( -f $filename) ) echo file doesn't exist
t2:eesoft>  if ( -d  /tmp ) echo the dir /tmp exists
```

ראו פירוט של השאילתות בדפי ההרצאות.

3.5. ערכים נומריים

לביצוע פעולות נומריות, יש להקדים את הסימן @ לפני כל שורה:

```
t2:eesoft>  set  k  = 5
t2:eesoft>  @ j = 5                    # the same as above
t2:eesoft>  @  s = $j % $k
t2:eesoft>  echo $s
0
t2:eesoft>  while ($k )
echo $k
@ k - -
end
5
4
3
2
1
t2:eesoft>
```

הערה: **תמיד** יש להשאיר **רווח** בין הסימן @ לבין הביטוי.

4. scripts

4.1 כתיבת script

ניתן לאגד קבוצה של פקודות c-shell לרצף בתוך קובץ אחד בר הרצה, קבוץ כזה יקרא script (בדומה לקבצי bat. ב – DOS) :

1. ראשית יש לרשום בשורה הראשונה של הקובץ את שני התוים "#!" , ואחריהם (באותה שורה) ה- full-path של תוכנית ה- c-shell עצמה, כמו כן מומלץ לרשום את האופציה f– :

```
#!/usr/bin/csh -f
```

(למתקדמים : שורה זו מייצגת את סוג ה-script, כלומר באיזה אינטרפרטר צריכים להשתמש על מנת לבצעו, והוא יכול להיות לדוגמא קובץ של tcsh או perl) .

2. יש למלא בשאר שורות הקובץ את הפקודות שרוצים לבצע.
3. ניתן לרשום הערות בין השורות : שורת **הערה** מתחילה ב- **#.**
4. בסיום כתיבת הקובץ , יש להפוך אותו לקובץ הרצה ע"י chmod :

```
t2:eesoft>  chmod +x  my_script
t2:eesoft>  my_script                  # runs my_script
```

הערה : ניתן לוותר על שלבים 4+1, אך בשביל להריץ קובץ script כזה, יש לקרוא לפקודה **source** : **source my_script**.

4.2 פרמטרים לקבצי script

ניתן להעביר **פרמטרים** ל-script, וניתן להתייחס לפרמטרים אלה בתוך קובץ ה-script :

- $0 – **הפקודה** עצמה
- $1 – הפרמטר **הראשון**
- $* - **רשימת כל הפרמטרים** הנתונים לפקודה (לא כולל $0)
- $#argv – **מספר הפרמטרים** הניתנים לפקודה (לא כולל $0)
- $argv[n] או $n – פרמטר **מספר n**

דוגמא: קובץ בשם print שתפקידו לשלוח להדפסה את הקבצים שהוא מקבל כפרמטרים:

```
#!/usr/bin/csh –f
foreach F ($*)
        echo printing $F
        lpr $F
end
```

דוגמת הרצה:

```
t2:eesoft>  print a.c b.c prog.c
printing a.c
printing b.c
printing prog.c
t2:eesoft>
```

4.3. דוגמא ל script

נתון קובץ בנק המכיל נתונים על כל הפעולות של הבנק. כל שורה בקובץ מייצגת פעולה אחת עם שם הלקוח, פעולה (הפקדה + והוצאה –), סניף, יום ושעה:

```
t2:eesoft>  cat bank
Dan Shilon      5000      Ramat-Aviv       1/1/1998 10:34
Dudu Topaz      10000     Herzliya-Pituah  1/1/1998 11:22
Haim Yavin      20000     Jerusalem        1/1/1998 15:18
Boris Lavva     –1000     Technion         1/1/1998 17:11
Dan Shilon      3000      Ramat-Aviv       14/1/1998 10:34
Dudu Topaz      11000     Herzliya-Pituah  15/1/1998 13:21
Haim Yavin      18000     Jerusalem        15/1/1998 16:11
Boris Lavva     –2000     Technion         16/1/1998 11:13
```

ברצוננו לכתוב script בשם client, אשר יקבל כפרמטר שם מלא של לקוח וידפיס את כל השורות של קובץ bank, בהן מופיע שם הלקוח וגם סה"כ היתרה בחשבונו. למשל, עבור Hayim Yavin נקבל:

```
t2:eesoft>  client "Hayim Yavin"
Haim Yavin      20000      Jerusalem      1/1/1998   15:18
Haim Yavin      18000      Jerusalem      15/1/1998  16:11
Total balance: 38000
```

שימו לב: רשמנו את השם המלה בתוך **גרשיים** " ", כך שה- script יוכל להתייחס אליו **כמחרוזת אחת** !!

נשתמש ב2- תוכניות script:

```
t2:eesoft>  cat client
#!/usr/bin/csh -f
if ($#argv != 1) then
        echo Usage: $0 client_name
else
        grep "$1" bank | calc_total
endif
```

גם כאן, רשמנו את $1 בתוך גרשיים, ע"מ ש- grep תחפש את המחרוזת המלאה !

השורה עם ה- grep שופכת את הפלט של ה- grep לתוך script שני שנקרא calc_total. ה- calc_total קורא שורות מהקלט הסטנדרטי שלו ומסכם את השדה השלישי הכל שורה:

```
t2:eesoft>  cat calc_total
#!/usr/bin/csh -f
# reading the first line
set line =  $<
set line = ($line)
@ sum = 0
while ($#line != 0)             #  $#line  == 0 means EOF
        @ sum = $sum + $line[3]
        echo $line
        set line = $<
        set line = ($line)
end
echo "Total balance: $sum"
```

הסימן המיוחד **$<** קורא שורה מהקלט, ומחזיר אותה כמחרוזת אחת .

לרב, מפצלים את שורה הקלט הנקראת ממחרוזת אחת למערך של מחרוזות, לדוגמא: **set line = ($line)** .

4.4. קבצי setup

קבצי setup הם קבצי script מיוחדים אשר מורצים אוטומטית ע"י המערכת בזמנים מסוימים, וכל משתמש יכול להחזיק קבצי setup משלו במדריך הבית שלו (home directory) . קבצים אלה הם:

- **.login** – מבוצע עם כל כניסה של משתמש לחשבון (login-ing)
- **.cshrc** – מבוצע בהתחלת כל c-shell. האופציה f– של ה csh מבטלת את ביצועו, מה שמאיץ את ריצת ה- script.

כיוון שקבצים אלה מופעלים בצורה אוטומטית בזמנים המצוינים לעיל, נהוג לאתחל בהם משתני סביבה חשובים, או להגדיר שמות חדשים לפקודות עם **alias** , למשל:

```
t2:eesoft>  more .login
set path = ($path /usr/local/gnu/bin)

t2:eesoft>  more .cshrc
alias ll  'ls –al'
alias m  more
```

5. Aliases

5.1 הגדרת alias:

ניתן להגדיר שם חדש לפקודה ע"י שימוש ב-**alias**:

```
t2:eesoft>  alias  del  rm
```

ואז ניתן להשתמש בפקודה del במקום rm:

```
t2:eesoft>  del file2
```

לקבלת כל ה aliases הנמצאים ב c-shell הנוכחי:

```
t2:eesoft>  alias
del rm
dir ls -l
t2:eesoft>
```

על מנת לדעת מה הפקודה המתאימה ל alias נתון:

```
t2:eesoft>  alias dir
ls –l
t2:eesoft>
```

5.2 מחיקת alias:

לביטול alias מתוך רשימת ה aliases של ה c-shell הנוכחי, משתמשים בפקודה **unalias**:

```
t2:eesoft>  unalias del
t2:eesoft>  alias
dir ls –l
t2:eesoft>
```

5.3 עקיפה זמנית של alias:

כשה c-shell מקבלת פקודה, היא מחפשת אותה בטבלת ה- aliases שלה לפני החיפוש ב- path.

המנגנון הזה מאוד חזק ומאפשר למשתמש לבנות לעצמו רמה של פקודות מעל פקודות ה unix הסטנדרטיות, למשל: הפקודה rm ב- unix מוחקת

קבצים בצורה מיידית, ואם מישהו רשם * rm בטעות, אזי הוא מחק את כל המדריך שלו , לא נעים !!

בשביל זה קיימת האופציה **–i** של rm , הגורמת לכך שעל כל קובץ שמנסים למחוק, ה rm תשאל את המשתמש אם הוא באמת רוצה למחוק:

ולכן להרבה משתמשים קיים ה **alias** הבא:

```
t2:eesoft> alias rm rm –i
t2:eesoft> rm x.o
rm: remove x.o (y/n)? y
```

וה- rm הנקרא בשורה השניה הוא ה alias rm !!

<u>מאידך</u>, לפעמים אני בטוח במה שאני הולך למחוק, ואני לא צריך את ה-alias , אך גם לא רוצה לבטל אותו לחלוטין, אלא פשוט לעקוף אותו חד-פעמית

```
t2:eesoft> \rm *.o
```

הוספת ה "\" לפני הפקודה מורה ל c-shell **לעקוף** את טבלת ה aliases, ולחפש את הפקודה אך ורק ב path.

5.4 <u>הפקודה which</u>

הפקודה **which** פקודה פנימית של ה c-shell שמזהה פקודה נתונה אם היא alias, פקודה פנימית של ה- shell או קובץ בר-הרצה:

```
t2:eesoft> which m
m:    aliased to more
t2:eesoft> which set
set:  shell built-in command
t2:eesoft> which grep
/usr/5bin/grep
```