בס"ד

הצגת מידע על משתמשים במערכת

בפקודה finger מציגה מידע על משתמשים במערכת
דוגמא:
finger –l coheni
בדוגמא הנ"ל האופציה l– מורה לפקודה finger להציג מידע מפורט (long) על המשתמש coheni .

בצעו finger amir
בצעו finger –m amir
בכל מיקרה אשר תרצו עוד מידע על הפקודה finger (כמו גם על כל פקודה) תשתמשו ב- man
בצעו finger –m moshe

בצעו את הפעולות הבאות:

cat > .plan
I have no plan for the next 3 year
(because …)
^D
chmod 755 .plan (למה זה נחוץ?)
finger –m <*login*>

who

הצגת כל המשתמשים הפעילים המחשב.

last –n <*number*>
מציג את מספר המשתמשים האחרונים שנכנסו למערכת.
last –n <*number*> <*login*>
מציג מתי המשתמש המצוין נכנס למערכת ב- n הפעמים האחרונות.

wc

ספירת מס' התווים,המילים והשורות בקובץ wc <-cwl> filename
wc file
יתן פלט שכזה:

56	195	3125
שורות	מילים	תווים

tail

הצגת מספר שורות מסוף הקובץ עיינו ב- man ע"מ לראות כיצד משתמשים בה.
בדומה ל- tail בדקו מהיא הפקודה head.

תצוגת ה- prompt

שינוי מבנה ה-prompt: set prompt="%n %m %~ %t %h : "
%n – login המשתמש.
%m – שם השרת.
%t – הזמן.
%h – היסטוריה.
%~ - באיזו ספריה אתם כעת, עם המסלול.

sort

מיון השורות בקובץ מסוים sort [file]

מיון: [r - ממיין הפוך, n –ממיין ע"פ ערך מספרי ולא לקסיקוגראפי, x.y + ממיין ע"פ התו ה- y בשדה ה- x]

cat filename | sort [-rn][+#.#]

הפקודה:

sort file

תציג את הקובץ ממוין לפי התחלת כל שורה.

sort –r file

תציג אותו הדבר, רק מהסוף להתחלה.

הפקודה: sort +2 file

תציג את הקובץ כשהוא ממוין לפי המילה השלישית בכל שורה.

(מתחילים למספר את מיקום המילים בשורה מאפס, כמו במערכים, ולא מאחד).

```
-rw-r--r--   1 myghaz   grad      41984 Jun 22 17:16 tasta.doc
drwxr-xr-x   2 myghaz   grad       1024 Oct 25  1998 S_P/
```

חיפוש קבצים בעץ הקבצים

פקודת חיפוש בעץ הקבצים: find directory –name *filename* –print

-directory הספריה ממנה מתחיל החיפוש.

- filename שם הקובץ המבוקש.

- –print לא חייב (אופציונאלי).

דוגמא:

בצעו את הפקודה: find ~/ –name *myfile* –print

בדוגמא זו תחילת החיפוש הוא מהספריה הראשית שלכם, מה שבדרך כלל עושים. אם אתם רוצים לעשות חיפוש מהספריה שבה אתם נמצאים ברגע זה אז במקום /~ צריך לכתוב (נקודה)

find . –name myfile *–print*

find . -name core -exec \rm {} \;

פקודה זאת מוחקת את כל קבצי ה-core שמופיעים בתתי ספריות של הספריה הנוכחית כולל הספריה עצמה.

בידקו מה פקודה זאת עושה:

find /usr/include/ -name 'e*' -exec echo {} start with e \;

pipe

pipe – מה זה?
זה "צינור" המעביר מידע מתכנית תכנית אחת לאחרת.
לדוגמא נרצה להעביר את הפלט של הפקודה ls –l לפקודה less כדי שהפקודה less תעבוד על הפלט. נניח שישנם הרבה מאוד קבצים ואם תעשו ls –l תראו רק את החלק התחתון של הרשימה.
כמו שלמדנו הפקודה less נותנת אפשרות דפדוף וכאן אנו "נדפדף" בפלט של הפקודה ls –l .
בצעו:

cd /temp (זו ספריה שבד"כ מכילה הרבה קבצים)
ls –l | less [שימו לב כי ה- pipe מסומן ב- '|' (הסימן מעל ה- \)]

בצעו:

finger | sort מה אתם רואים?
finger | sort > users.sorted בידקו מה קרה, האם נוצרו קבצים חדשים?

ביטויים רגולריים בשמות קבצים:

אם תבנית שם קובץ מתאימה ליותר משם קובץ אחד, ה- shell יחליף אותה **בכל** השמות המתאימים (למשל ls *) . אם אין אף התאמה לתבנית (למשל ספריה ריקה) הפקודה לא תתבצע בכלל, ואם דבר כזה קורה בסקריפט, הסקריפט יעוף עם שגיאה.

1. לכל תו רגיל (או רצף תווים) – מתאים לאותו חלק של שם קובץ שם הם מופיעים.
דוגמא (טריוויאלית):
ls –l /usr / include / stdio.h
יציג את פרטי הקובץ הנ"ל.

2. כוכבית - * : מתאים לרצף של תווים כלשהם.
דוגמאות:
ls –l /usr / include /*.h
יציג את פרטי כל קבצי ה- header בספריה הנ"ל.

ls –l /usr / include / std*.h
יציג את פרטי כל קבצי ה- header (הכותרת) בספריה הנ"ל ששמם מתחיל ב- std .

grep –i moshe * * /*
יחפש את המחרוזת moshe בכל הקבצים בספריה הנוכחית ובתת ספריות שלה (רמה אחת לעומק בלבד).

3. סימן שאלה - ? : מתאים לכל תו בודד.
דוגמאות:
ls –l / usr / include /*.?
יציג את פרטי כל הקבצים שהסיומת שלהם היא תו בודד
(לא יציג קבצי cc. ותת ספריות (שאין להם סיומת בכלל)).

הערה: כוכבית (או סימן שאלה) בתחילת שם קובץ לא לתופשת שמות קבצים שמתחילים בנקודה (.) – אלה קבצים שהם hidden .

דוגמא:

grep setenv *

לא יחפש את המילה stetenv בקובץ "login" .

4. סוגריים מסולסלים– {str1, str2, str3} מתאימים למופע של כל אחת מהמחרוזות שבסוגריים המסולסלים.
דוגמא:

cp / usr / include /{stdio, stdlib}.h /tmp

יעתיק את שני הקבצים /usr / include /stdio.h ו- / usr / include /stdio.h לספריה /tmp .

5. ~ (tilda) הוא קיצור דרך ל- home directory של המשתמש. moshe~ הוא קיצור דרך לספריה של moshe.
דוגמא:

cp ~moshe/ .cshrc ~/

יעתיק את הקובץ cshrc . מהספריה של moshe לספריה שלי.

<u>ביטויים רגולריים ב- grep סטנדרטי:</u>

1. כל תו שאינו תו מיוחד מתאים לעצמו.
דוגמא:

grep moshe *

מחפש את המחרוזת moshe בכל הקבצים בספריה הנוכחית.

2. הסימן ^ בתחילת ביטוי מצמיד את הביטוי לתחילת השורה. הסימן $ בסוף שורה מצמיד את הביטוי לסוף שורה.
דוגמאות:

grep ^moshe *

יחפש את כל השורות שמתחילות במילה moshe .

grep david$ *

יחפש את כל השורות המסתיימות במילה david.

grep ^rachel$ *

יחפש את כל השורות שבהן מופיעה המילה rachel בלבד!.

grep ^$ *

יחפש את כל השורות הריקות.

3. "." (נקודה) מתאימה לכל תו פרט לתו newline (\n).
דוגמאות:

grep . *

יחפש את כל השורות שאינן ריקות (יש בהן תו כלשהו).

grep .avid

ימצא את השורות בהן מופיע "david"או "ravid" וכו'.

4. רצף סימנים בסוגריים מרובעים [] מתאימים לתו אחד מתוכם. אם הסימן ^ מופיע ראשון בתוך הסוגריים המרובעים התבנית מתאימה לכל תו פרט לשאר התווים בתוך ה- []. הסימן "-" (מינוס) בין שני תווים מציין רצף תווים בין שני התווים שמלצדדיו.
דוגמאות:

grep [dr]avid *

יחפש את השורות שבהן מופיע "david" או "ravid" אך לא "savid".

grep [1-9][0-9] 4

יחפש את השורות שבהן מופיעים רצפים של שלוש ספרות
כאשר הראשונה אינה אפס והאחרונה היא 4.

grep ^[^a-zA-Z]

יחפש את כל השורות שלא מתחילות באות אנגלית (קטנה או גדולה).

5. אחרי כל אחד מהנ"ל יכולה לבוא הכוכבית "*" וביטוי החדש מתאים לרצף של אפס או יותר מופעים של מה שבא לפני הכוכבית. יש לשים את הביטוי במרכאות כדי לא לבלבל את ה- shell .
דוגמאות:

grep "da*d" *

ימצא את השורות שבהן מופיע "dad", "daaaaad" , אבל לא "dadad" כי הכוכבית פועלת רק על הדבר האחרון שהיא רואה וזה "a" .

grep "[0-9][0-9]*4"*

ימצא את כל השורות שבהן מופיעים מספרים המסתיימים ב-4 והם לא 4 (יש לפחות עוד ספרה בהתחלה).

6. אם אחרי כל אחד מהביטויים 1-5 באים סוגריים מסוסלים עם מספר בתוכם, הביטוי החדש יחפש את מה שיש לפני הסוגריים המסולסלים מספר פעמים כמו שרשום בתוך הסוגריים המסולסלים. צריך להקדים את הסוגריים עם backslash, כפי שמופיע בדוגמאות למטה.
דוגמאות:

grep "me\{2\}t" *

ימצא את "meet" אבל לא את "met" .

grep "^m[aouei]\{2\}" *

ימצא את כל השורות המתחילות באות "m" ואחריה שתי תנועות (vowels).

7. ניתן לשים ביטוי רגולרי בסוגריים כדי להתייחס אליהם בהמשך. גם כאן צריך להקדים את הסוגריים ב- backslash. ההתייחסות למה שמצאנו בסוגריים היא על ידי \1. הסבר בדוגמא.
דוגמא:

grep "\([^aouie][aouie]\)\1"

ימצא מילים מסוג
"papa", "vivid", "banana", "cucumber", "homomorphism",

כלומר כאלה שחוזר בהן הצרוף "עיצור- תנועה" פעמיים. לא יתאימו למשל מילים כמו "sota" שכן הצרוף הראשון והשני שונים, so != ta.

8. סימנים מיוחדים "\>" ו- "\<" מתאימים לתחילת מילה ולסיומה בהתאמה.
דוגמא:
grep "\<[a-zA-Z]\{4\}\>"
ימצא את כל השורות שיש בהן מילים בנות 4 אותיות.

- כתבו פקודה שסופרת את מספר הספריות בספריה הנוכחית (רמז הסתכלו על הפלט של ls –l)

AWK

awk מאפשרת לנו לעשות מניפולציות על קלטים/קבצים.
$0 - כל השורה.
$1 - השדה הראשון בשורה, כאשר שדה הוא **בעיקרון** מילה. המפריד בין השדות זה רוח (ואם לא).
$2 - השדה השני בשורה. ועל זו הדרך לכל $x.
NF- מספר השדות בשורה (לכן $NF – השדה האחרון בשורה.)
NR- מספר השורה הנוכחית.

דוגמאות:

- awk '$1 < $2 {print $0, $1/$2}' file1 > file2

הסבר קל: החלק הראשון הוא תנאי אם השדה הראשון קטן מהשדה השני אז תדפיס את כל השורה שזה מתקיים בה ובנוסף תדפיס את תוצאת החילוק השדה הראשון חלקי השדה השני. העיבוד הזה מתבצע על הקובץ file1 והפלט אינו עובר למסך כי אם אל הקובץ file2.

עוד דרך לעשות אותו הדבר cat file1 | awk '$1 < $2 {print $0, $1/$2}' > file2

- awk '{ print $2, $1 }' file

עבור על הקובץ file והדפס את שני השדות הראשונים בסדר הפוך.

- awk 'length > 72' file

הדפס את השורות אשר אורך השורה גדול מ- 72 תווים.

- awk '{print length($2)}' file הדפס את אורך השדה השני.

בצעו את הפקודות הבאות:

```
cat > myfile
1
2
3
^D
cat myfile | awk '{ s=s+$1 ; print s}'
```

- `ls *.cpp | awk '{print "mv "$0" ../cppDir/"$0".old"}' | tcsh`

מעביר את כל הקבצים של C++ לספריה cppDir כאשר בסוף משרשר לשם סיומת של old. עד ל- pipe השני כל מה שהוא עושה הוא רק "בונה" מחרוזת מהמבנה הבא mv file1.cpp file1.cpp.old ואז דרך ה- pipe הוא מעביר ל- shell ואז ה- shell מבצע את ההעברה.
שימו לב שבדרך זו אפשר להפעיל פקודות שונות של ה- shell ע"י awk.

עוד דוגמה:

```
last -n 10 | awk '{print "finger -m " $1}' | sh | grep "In real life" | awk '{print $7 " " $NF}'
```

מה פקודה זאת עושה??

- cat *filename* | awk '{print $2,$1*3,($3+8)%10}'

בדקו מה עושה פקודה זו יש לתת את הדעת על הקובץ *filename* (קובץ מספרים וכו').

- **כתבו פקודה שתדפיס את השמות הפרטים של האנשים שנמצאים עכשיו על המערכת אך לא תחזור על שם פעמיים.** (רמז: יש פקודה uniq)

SED

Stream editor המבצע מניפולציות בתוך טקסט בקבצים.

- cat *filename* | sed ' s/str1/str2/g'

s - מחליף (substitute) את str1 ב- str2.
g – גלובלית על כל הקובץ.

alias

ביוניקס ניתן ליצור "שמות חיבה" (קיצורים) לפקודות ע"מ להקל על המשתמש. לדוגמא לפקודה ls –l יש קיצור והוא ll. כדי ליצור קיצור לפקודה משתמשים בפקודה alias.

לדוגמא:

alias dir "ls –l"
alias shalom "kill –1 –1"
alias up "cd .."
alias cd "cd \!*;pwd;ls"
alias myecho "echo It's mine \!*"

(שימו לב מה עושה *!\)

כעת ניתן להתייחס אל הקיצורים של הפקודות (למשלup)כאל פקודות לכל דבר.

dir יעשה בדיוק מה ש- ls –l עושה (רשימת קבצים עם מידע מורחב).

shlom תוציא אתכם מהמחשב.

up תעביר אתכם ספריה אחת למעלה.

- כתבו קיצור (del_backup) אשר ימחק את כל קבצי ה backup בספריה הנוכחית ותתי הספרויות שלה (קובץ backup מתחיל ב ~).
- כתבו קיצור דרך (iavg) אשר יחשב את ממוצע גודל הקבצים בספריה הנוכחית
- כתבו קיצור דרך (davg) אשר יחשב את ממוצע גודל קבצים בספריה נתונה בעת הפעלה (רמז בעזרת *!\) דוגמה: לשורת הפעלה davg /usr/include/.

הפקודה unalias <*command*> מבטלת את הקיצור, unalias * מבטל את כל הקיצורים