**Default(Systemove prikazy, pouzivatelske, other)**

**SHORTCUTS**

**ctrl+c** = zrusi hocijaky proces

**ctrl+s** = pause

**ctrl+q** = continue

**ctrl+d** = vypise vsetky moznosti

**ctrl+a** = na zaciatok riadku

**ctrl+e** = ist na koniec riadku

**alt + b** = back one word

**alt + f** = forward one word

**ctrl + xx** = toggle between the start of line and current cursor position

**ctrl + k** = zrusi vsetko za kurzorom

**ctrl+w** = zrusi text v prikaze

**passwd** = zmena hesla

**man** = manual a nasleduje prikaz

**sipky** hore/dole ukaze prikazy z minulosti

**clear** = vycistenie

**date** = vypise cas

**apropos** - ked nevieme ako sa vola nejaky prikaz, tak mozme k nemu pridat klucove slovo

**| -** pajpa, pomocou ktorej sa presunemie do dalsieho prikazu

Pouzivatelia

**who** - vypise kto je lognuty(long name, tty name, cas lognutia, hostname)

**w** - display who is logged into the Linux and Unix-like server, and what they are doing at command execution time

**finger -** vypise udaje prihlasenych pouzivatelov

**users** - vypise prihlasenych pouzivatelov do riadku

**write** - pisanie niekomu

**mesg n** - zakazat napisanie

**whoami** - vypise moj login

**id** - info o mne

**last** - posledne prihlaseny pouzivatelia

**history -** vypise historiu prikazov

**umask** - (nastavenie prav nasledujucich vytvorenych suborov, v trojici pouzivatelov(user,group,other)

     0 - rwx, 1 - rw\*, 2 - r\*x, 3 - r\*\*, 4 - \*wx, 5 - \*w\*, 6 - \*\*x, 7 - \*\*\*

     umask - vypise aktualne nastaveny umask

     umask <trojica cisel> - nastavenie prav pre kazdy typ pouzivatela v binarnom formate

**chmod <mod> <subor...>** - modifikuje pristupove prava zadanych suborov

mody: **<ugo><+-><rwx>**

**u**ser/**g**roup/**o**ther **pridat**/**odobrat r**ead/**w**rite/e**x**ecute

chmod o+x - ostatnym prida pravo na spustenie     (STANDARD)

chmod 700 - mne da vsetko, ostatnym nic     (BITOVO)

**Praca s adresarmi**

**ls** - vypise co mame v adresari

**ls -l**  (ukazu sa subory s viac informaciami)

**ls /**   - absolutna cesta (root)

pri vypise zacinaju pristupove prava.. ide sa po trojiciach..

1) trojica urcuje prava vlastnika, (moze citat(r) pisat(w), pustat(x)),

2) trojica - prava skupin,

3) trojica je pre vsetkych ostatnych

     potom je tam jednotka - pocet liniek ktore ukazuju na subor (2 + pocet suborov v subore)

     cislo pred datumom je - velkost

     cas poslednej modifikacie

**ls** -la (ukaze aj skryte subory)

**cd** - meni directory kde sme

     napr cd .. sa dostaneme vyssie o jeden adresar

     samotne cd nas dostane do nasho adresara

**cd ~<login\_name>** - dostanie sa docudzieho adresara

**pwd** - vypise kde sa nachadzam

**mkdir <meno adresy>** **<dalsi..>**- vytvorenie adr.

**rmdir <meno adresy>** **<dalsi..>**- vymazanie adr.

**Praca so subormi**

**>** - napise do suboru, prepise teda

**>>** - napise na koniec suboru

**cp** - <co chcem kopirovat> <kam chcem kopirovat>)

**mv -** <co presuvam...><kde presuvam> (premenuje sa ak neexistuje)

**rm** - zmazanie suboru

**touch** - vytvorenie suboru

**less -** lepsie nez more (umoznuje pohyb v subore aj spatny pohyb a ma viac moznosti) [q na vypnutie, h na help]

**cmp <file1> <file2> -** porovnavanie dvoch suborov - vypise kde je prvy znakovy rozdiel

**diff** - V najjednoduchsom pripade diff porovnava obsah dvoch suborov from-file a to-file. Ak from-file je adresar a to-file nie je, tak diff porovnava subor v adresari from-file, ktory ma rovnaky nazov ako subor to-file. Ak oba from-file a to-file su adresare, diff porovnava korespondujuce subory v oboch adresaroch.

**file <subor..>** - zisti aky je to typ suboru

**Vyhladavanie suborov (find)**

**find** - this command lists out all the files in the current directory as well as the subdirectories in the current directory ***find location comparison-criteria search-term*** -

**find <nic-in current dir> / <directory(moze byt ich aj viac, oddelene medzerou>** - Lists out all files by default

**find <directory> -name "abc.txt"** - searches for files by their name

**find <directory> -name "\*.php"** - searches for files which have .php in the end

**find <directory> -iname "\*.Php"** - searches for files which have .php or .Php in the end (nezalezi teda na velkosti pismen)

**find <directory> -maxdepth 2 -name "\*.php""** - searches will not go lower than 2 level deep (1. level je miesto, kde sa nachadzaju subory v <directory>)

**find <directory> -not -name "abc.txt"** - searches for files which dont have name "abc.txt"  [not je to iste ako vykricnik]

**find <directory> -name 'abc\*' ! -name '\*.php'** - looks for files that begin with abc in their names and do not have a php extension

**find <directory> -name '\*.php' -or -name '\*.txt'** - search for files ending in either the php extension or the txt extension

**find <directory> -type f -name "abc\*"** - finds only files that begin with abc

**find <directory> -type d -name "abc\*"** - finds only directories that begin with abc

**find ~ -type f -name ".\*"** - vyhlada skryte subory, to ~ ukaze celu cestu k adresaru

viac na <http://www.binarytides.com/linux-find-command-examples/>

**-exec command ; (Pri FINDE)**

Execute command; true if 0 status is returned.  All following arguments to find are taken to  be  arguments  to  the command until an argument consisting of `;' is encountered.

The string `{}' is replaced by the current file name being processed everywhere it occurs in the arguments to the command, not just

in arguments where it is alone, as in some versions of find.  Both of these constructions might need to be  escaped  (with a `\') or quoted to protect them from expansion by the shell.  The command  is  executed in the starting directory.

**Vyhladavanie textu v suboroch (grep)**

**grep** - vyhladavanie v texte na zaklade patternu (stdin + file)

**grep <co hladam> <file> / <vystup> | grep <co hladam>**

**grep -i "boo" -** ignoruje velkost pismen

**grep -w "boo" <file>** - hladanie konkretneho slova (bez post/prefixov)

**grep -E -w "word1|word2" <file> -** vyhladavanie 2 slov (extended)

**grep -c "word" <file>** - vypise pocet najdenych slov

**grep -n "word" <file>** - ocisluje kazdy najdeny riadok cislom

**grep -v "word" <file>** - vypise len tie riadky ktore neobsahuju 'word'

**praca s obsahom suboru**

**Praca s vypisom**

**cat -** (stdin-file)

**head -** (stdin-file) **(head -n 5      - vypisanie prvych 5 riadkov)**

**tail**- opak headu (ide od spodu)

**echo** "tralala la" > subor2 (presunieme tento vystup do suboru.. ked neexistuje tak sa automaticky vytvori)

     echo tralala - vypise na vystup

**rev -** reverse lines of a file (stdin + file) - pise znaky od konca riadku

**bc -** matematicke operacie (<http://www.folkstalk.com/2012/09/bc-command-examples-in-unix-linux.html>)

**wc -** počíta znaky, riadky a slová (stdin + file) znaky(wc -c), riadky(wc -l), slova(wc -w) alebo defaultne vsetko

**sort -** Usporiadanie riadkov (stdin + file)

**uniq -** eliminuje identicko-susedne riadky(stdin + file). Ak su na konci zadane subory, odstrani duplicitne susedne riadky z file1 a vypise vystup do file2

**-c     -** spocita duplicity identickych riadkov

**-d** - zobrazi len duplicitne riadky

**-u** - zobrazi len neduplicitne riadky

**-i**- ignoruje sa velkost pismen

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**cut -** vyberie zvolene stlpce z textu (stdin + file)

***cut -c list[súbor...]***

***cut -f list [-d delim] [-s] [súbor...]***

list špecifikuje stĺpce, ktoré sa majú vybrať

-c stĺpce sú chápané ako jednotlivé znaky

-f stĺpce sú chápané ako znaky oddelené tabulátormi

-d delim určuje oddeľovací znak, ktorý sa použije namiesto tabulátora

**cut -c 2,15-20,25-     -** vypise sa v kazdom riadku druhy znak, vsetky znaky od 15-20, vsetky znaky od 25

**cut -d <delimiter> -f<cisla>** (dobry priklad s delimitrom -> last | cut -d ':' -f1 {a potom} f2

**-s** - vypise riadky iba tie, ktore obsahuju ten delimiter

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**tr =** translate characters, (stdin)

***tr <strToChange> <stringPattern> tr Bran Petr -> ked najde a, nahradi ho t***

**tr -d <stringPattern>** = vymazu sa vsetky znaky znakmi zo <stringPattern>

**<strToChange><stringPattern>** = nahradza vsetky znaky okrem znakov uvedenych v <strToChange>

**tr -s** = ak sa nachadza viac tych istych znakov v strToCharge po sebe ktore sa maju zmenit na znaky zo stringPattern, nahradia sa len jednym znakom

**tr -cd [:digit:]** - remove all chars except digits

**tr “[:lower:]“ “[:upper:]“**

MweFd23

*MWEFD23*

**tr -cd “[:print:]“ < subor**

- odfiltruje netlačitelné znaky zo súboru

**tr -cs “[:alpha:]” “\n“ < subor**

- kazdy znak ktory nieje [:alpha:] sa zmeni na \n (vsetky tieto znaky susediace sa zlucia)

- každé slovo na jednom riadku

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ZNAKY**

**Úvodzovky sa používajú na uvádzanie hodnôt obsahujúcich medzery alebo špeciálne znaky, na oddelenie premenných a na označenie príkazového výstupu.**

**' '** -> odstrani spec. zmysel znakov okrem ' a !

**" "** -> odstrani spec. zmysel znakov okrem $, `, " a !

**\c** -> odstrani spec. zmysel tomu c

**`command`** -> (backticky) nahradi standardnym vystupom prikazu command <https://www.youtube.com/watch?v=VOOeXV4HYSA>

**Metaznaky - ak chcem aby stratili vyznam, musi byt pred nimi spatne lomitko \**

**<SPACE>**-> ako oddelovac prikazov

**<TAB>**-> ako oddelovac prikazov

**<RETURN>**-> ukoncovac prikazu

**$** -> identifikator premennej

**?** -> najde hocijaky znak raz

**\*** -> matches any string, including the null string

**[], {}, ~** -> pouzivane pre expanziu mien suborov

**<, >, &, !** -> presmerovanie

**!, ^** -> historia

**|** -> pipe

**;** -> ukoncovac prikazu

**(, )** -> skupina prikazov

**\, ', "** -> uvodzovkovanie

**`** -> nahrada prikazov

**&** -> vykonavanie na pozadi

[**PRÍKLADY PRÍKAZOV**](http://osa.fiit.stuba.sk/os/prirucka_csh/ch01/030-035.html#Heading22)

**REGULAR EXPRESSION is a concept of matching a pattern in a given string. (awk, ed, egrep, grep, sed, a vi)**

<https://www.gnu.org/software/findutils/manual/html_node/find_html/egrep-regular-expression-syntax.html#egrep-regular-expression-syntax>

<https://www.gnu.org/software/findutils/manual/html_node/find_html/grep-regular-expression-syntax.html>

<http://www.linuxnix.com/regular-expressions-linux-i/>

<http://www.linuxnix.com/grep-command-regular-expressions-examples-ii/>

**^** - match beginning of the line

**$** - match end of the line

**\*** - match 0 or more occurrences of previous character

**.** - match any character

**/<word/>** - actual word finding, (JE TREBA DAVAT BACKSLASH aj pre grep -E)

**/<** - zaciatok slova

**\>** - koniec slova

**[]** - match range of chars, just single occurrence

**[abc]** - najde akykolvek znak medzi zatvorkami

**[a-cx-z]** - najde vsetko medzi a-c, od x po z

**[^a-c]** - negovanie vsetko medzi a-c

**Interval regular expressions** je treba pripojit prepinac -E **(extended grep)**

a**+** = 1 az n (one or more occurrences of previous character)

a**?** = 0 alebo 1 (0 or 1 occurrence of previous character)

b**|**e - b alebo e (either character)

**{n}** - n of occurrence of previous character (po sebe iducich)

**{n,m}** - n to m occurrences of previous character

**{m, }** - m or more occurrence of previous character

**{, m}** - vyraz sa opakuje najviac m-krat

**()** - match a group of characters (grep -E "(\>[\.\!\?].\*){3}" ccero10.txt  (3 krat ukoncenie vety v riadku)

**preddefinované triedy znakov:**

[:alnum:] - same as [0-9A-Za-z]

[:alpha:] - same as [A-Za-z]

[:blank:] - blank characetrs: space and tab

[:digit:] - Digits: 0 1 2 3 4 5 6 7 8 9

[:lower:] - same as [a-z]

[:upper:] - same as [A-Z]

[:punct:] -  ! " # $ % & ' ( ) \* + , - . / : ; < = > ? @ [ \ ] ^ \_ ` { | } ~.

[:space:] - tab, newline, vertical tab, form feed(advance downward to the next "page", carriage return(vratenie sa na zaciatok.. "\r", and space

[:xdigit:] - hexadecimal digits

[:print:] - printable chars: [:alnum:], [:punct:] and space

[:graph:] - graphicla chars: [:alnum:] and [:punct:]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**REGULÁRNE VÝRAZY PODROBNE**

**Reťazce**

|  |  |  |
| --- | --- | --- |
| **Regulárny výraz** | **Vysvetlenie** | **Príklady** |
| /ivan/ | označuje *ivan* | ivan, ivanovi, odivanama |
| /ivan/ | označuje *ivan* | ivan, ivan's |
| /ivan ma/ | označuje *ivan ma* | ivan ma, ivan malo |
|  |  |  |

**Špeciálne znaky**

**Bodka**

|  |  |  |
| --- | --- | --- |
| **Regulárny výraz** | **Vysvetlenie** | **Príklady** |
| / .alk/ | označuje všetky reťazce, ktoré obsahujú medzeru <SPACE> nasledovanú ľubovoľným znakom, ktorý nasledovaný reťazcom *alk* | will talk, may talk |
| /.ing/ | označuje všetky reťazce, ktore obsahujú, ale nezačínajú na *ing* | singing, ping, before inglenook |
|  |  |  |

**Hranaté zátvorky []**

|  |  |  |
| --- | --- | --- |
| **Regulárny výraz** | **Vysvetlenie** | **Príklady** |
| /[bB]ill/ | označuje reťazce začínajúce znakom *b* alebo *B* nasledovaným reťazcom*ill* | bill, Bill, billed |
| /t[aeiou].k/ | oznčuje reťazce začínajúce znakom *t* nasledovaný samohláskou, ľubovoľným znakom a znakom *k* | teak, talkative, took, tanker |
| /number [6-9]/ | označuje reťazec *number* nasledovaný medzerou a číslom z rozsahu 6 až 9 | number 6, number 8, number 9 |
| /[^a-zA-Z]/ | označuje znak, ktorý nie je písmenom | 1, 7, @, ., } |
|  |  |  |

**Hviezdička \***

|  |  |  |
| --- | --- | --- |
| **Regulárny výraz** | **Vysvetlenie** | **Príklady** |
| /ab\*c/ | označuje reťazec, ktorý začína znakom *a* nasledovaný nula alebo viacerými inými znakmi *b* nasledovanými znakom *c* | ac, abc, abbc, abbbc |
| /ab.\*c/ | označuje reťazec, ktorý začína písmenami *ab* nasledovaný nula alebo viacerými znakmi nasledovanými znakom *c* | abc, abxc, ab45c, ab 756.345 c |
| /t.\*ing/ | označuje reťazec, ktorý začína znakom *t* nasledovaný nula alebo viacerými znakmi nasledovanými reťazcom *ing* | thing, ting, thought of going |
| /[a-zA-Z ]\*/ | označuje reťazce zložené iba z písmen a medzier | ľuboľný reťazec bez čísel a interpunkčných znamienok |
| /(.\*)/ | označuje najblhší reťazec ako je možné medzi *(* a *)* | (this) a (that) |
| /([^)]\*)/ | označuje najkratší reťazec ako je možné, ktorý začina *(* a končí *)* | (this) (this and that) |
|  |  |  |

**Strieška ^ a dolár $**

|  |  |  |
| --- | --- | --- |
| **Regulárny výraz** | **Vysvetlenie** | **Príklady** |
| /^T/ | označuje *T* na začiatku riadku | To je ono. |
| /^+[0-9]/ | označuje znamienko plus nalsedované číslom na začiatku riadku | +5 45.72 |
| /:$/ | označuje dvojbodku na koci riadka | ......ahojte: |
| /^$/ | označuje prázdny riadok |  |
|  |  |  |

**Spätné lomítko \**

|  |  |  |
| --- | --- | --- |
| **Regulárny výraz** | **Vysvetlenie** | **Príklady** |
| /end\./ | označuje všetky reťazce, ktoré obsahujú *end* nasledovaný bodkou | pretend.mail, the end., send |
| /\/ | oznčuje spätné lomítko \ | \ |
| /\\*/ | označuje hviezdičku \* | \*.c, hviezdicka (\*) |
| /\[5\]/ | označuje reťazec [5] | bolo pat [5] |
| /and\/or/ | označuje and/or | and/or |
|  |  |  |

**Pravidlá**

**Regulárne výrazy vždy označujú najdlhší možný reťazec !**

Napr. reťazec:*This (rug) is not what it once was (a long time ago), is it?*

Regulárny výraz: */Th.\*is/*

Označí: *This (rug) is not what it once was (a long time ago), is*

a výraz: */(.\*)/*

označí: (rug) is not what it once was (a long time ago)

ale výraz: */([^)]\*)/*

označí: *(rug)*

[**which** - Najdenie cesty k vykonatelnemu suboru](http://osa.fiit.stuba.sk/os/prehlad_prikazov/sys_prikazy/which.htm)

[**whereis** - Lokalizacia programov](http://osa.fiit.stuba.sk/os/prehlad_prikazov/sys_prikazy/whereis.htm)

[**tee** - Rozdelenie rury (pipe)](http://osa.fiit.stuba.sk/os/prehlad_prikazov/sys_prikazy/tee.htm)

[**Prikazy pre spravu suboroveho systemu**](http://osa.fiit.stuba.sk/os/prehlad_prikazov/sprava_subor_sys/index06.htm)

[**Prikazy pre pracu s procesmi**](http://osa.fiit.stuba.sk/os/prehlad_prikazov/procesy/index07.htm)

<http://osa.fiit.stuba.sk/os/prirucka_csh/ch01/006-010.html#Heading7>

**PRAKTICKE**

**2) SEMINAR**

**1)** kolko <niecoho> je v directory

ls | wc -l

**2)** kolko suborov ma tento adresar /home/os/pocitacove/csh

/home/OS/pocitacove/csh | ls | wc -l

**3)** kolko obycajnych suborov obsahuje adresar /home/os/pocitacove/csh

find <adresa>/<nic, ked som v tej adrese> -type f | wc -l     [43]

**4)**kolko suborov zacinajucich na pismeno p obsahuje adresar /home/os/pocitacove/csh

find -name "p\*" | wc -l

**5)**kolko riadkov sa zacina malym pismenom home/os/predtest/vstupy/ccero10.txt

grep -c '^[a-z]' ccero10.txt - naj prikaz

**6)** kolko neprazdnych adresarov (do hlbky) v tom istom adresari

find <adresa> -not -empty -type d

**7)** zistite pocet adresarov a pod-adresarov ale iba do hlbky 3 uroven [229]

find <adresa> -maxdepth 3 | wc -l

**8)** pocet obycajnych suborov a symbolickych liniek v adresari predtest[1808]

find -type f -or -type l | wc -l

**9)** najdite kolko znakov je na prvom riadku v subore ccero [65]

cat ccero10.txt | head -n 1 | wc -c

**10)**kolko netextovych obycajnych suborov je v adresari predtest

find -type f -not -name '\*.txt' | wc -l

**3) SEMINAR**

**2)** kolko studentov zacinajucich studiu v roku 2014 ma na serveri osa konto (/etc/passwd)

grep -c "^[a-z]\+14" /etc/passwd              [114]     -na zaciatku je male pismeno a slovo musi koncit 14

**3)** Kolko roznych pouzivatelov sa pripojilo do siete ynet.sk za poslednu dobu (od prveho oktobra)

chceme aby bolo ynet.sk na konci 3. stlpca

last | grep "^[a-z0-9]\+[[:space:]]\+tty..[[:space:]]\+.\*\.ynet\.sk\>" | cut -d ' ' -f1 | sort | uniq | wc -l

last | grep -E "^[a-z0-9]+[[:space:]]+tty..[[:space:]]+.\*.ynet.sk" | cut -d ' ' -f1 | sort | uniq | wc -l

last | grep "\.ynet\.sk\>" | cut -d ' ' -f1 | sort | uniq | wc -l     (ale asi by to nebolo uplne presne)

**4)**/home/OS/predtest/vstupy/ccero10.txt -> na kolkych riadkov je koniec vety .!.? za nejakym slovom

grep -c "\>[\.\!\?]" ccero10.txt

**5)** v adresari home/OS/predtest najdite aka je najvacsia velkost adresara predtest

find -type d -exec ls -ld {} \; | sort -n | cut -d ' ' -f8 | sort -n | tail -1

**4) Seminar**

home/os/literatura\_zakladna/joe.txt/vi.txtgre

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

cd ~silvia /konzultacie2

Ked pri hladani maxim alebo minim viacero suborov (adresarov, slov,

pismen, ...) splna podmienku, treba vypisat vsetky, zoradene podla abecedy

a oddelene jednou medzerou.

MOJA POZNAMKA: pozrite si vsetci /home/OS/literatura\_zakladna-FAQ\_testy

pozrite si zakladne prikazy co mate ovladat pred testom, je to na ose

naucte sa robit s editorom - kopirovanie z terminalu don bude na prostredi testu podobne ako mate na windowsoch, ale s rozdielom, ze nebudete kopirovat pravym tlacidlom mysi, ale kolieskom

taktiez na teste nepouzivajte numericku klavesnicu v editore, myslim, ze tam nefunguje spravne

naucte sa kopirovat bloky textov, skratky na rychle presuvanie posubore...

precitajte si hlavicku na teste - je rovnaka aj tuto

======================================================================

PRIKLAD 1

Kolko obycajnych suborov sa nachadza v adresari

/home/OS/predtest/vstupy-2/test2

VYSLEDOK 200

RIESENIE\_ZACIATOK

**find -type f | wc - l**

*pouzijete find s prepinacom, ktory vam najde obycajne subory a nasledne spocitate riadky*

RIESENIE\_KONIEC

======================================================================

PRIKLAD 2

Kolko znakov ma obsah premennej MANPATH ?

**echo -n $MANPATH | wc -c**           (-n zaruci, ze neprecita \n)

pozor na to, ze otazka je najst znaky premennej manpath, prikaz echo pridava na koniec znak noveho riadka, ktory nie je obsahom premennej, odstranite ho prepinacom, ktory prikaz echo ma

======================================================================

PRIKLAD 3

Prehladajte adresar /home/OS/pocitacove/csh a jeho podadresare a zistite kolko obycajnych suborov v nich

nema priponu.

**find /home/OS/pocitacove/csh -type f -not \( -name ".?\*?\*" -or -name "?\*.?\*" \) | wc -l**

pouzite prikaz find s prepinacom na zobrazenie obycajnych suborov a nasledne pouzite pattern co ma find (nie su to regulare vyrazy) spolu s prepinacom, ze hladate meno toho suboru s tym patternom, kde odstranite subory co maju priponu

======================================================================

PRIKLAD 4

V adresari /home/OS/predtest/vstupy-2/test3 a jeho podadresaroch najdite obycajny subor, ktory bol najmenej davno modifikovany a uvedte jeho meno. Ak je takych suborov viac, uvedte vsetky mena usporiadane podla abecedy.

**find /home/OS/predtest/vstupy-2/test3 -type f -printf "%T@ %f\n" | grep**

**`find /home/OS/predtest/vstupy-2/test3 -type f -printf "%T@ %f\n" | sort -n | cut -d " " -f1 | tail -n 1`**

**| cut -d " " -f2**

taktiez find, obycajne subory, a vypiste si nim casy modifikacie vo formate (pomocka @ - ktory vam napise pocet sekund od roku vzniku unixu 1.1.1970)

to si pekne zosortujete (pozor na to, ze chcete len meno suboru a nie cestu, preto si zvolte spravne vo finde co vam ma vypisat)

potom pride ta zabava, vsimli ste si, ze su tam tri s rovnakym casom najblizsie modifikovane a potrebujeme vypisat vsetky

preto pouzite napr subshell a nim si najdite este raz tie naposledy modifikovane, vygrapujte si v nom to cislo co vam urcuje ten pocet sekund pre tie najnovsie tri

tym dostanete vystup zo subshellu toto cislo a to si poslete do grepu povodneho shellovskeho prikazu co ste mali a z toho vystupu si vlastne vygrepujete tie tri najnovsie

======================================================================

PRIKLAD 5

Aky je sucet vsetkych cisel, ktore sa vyskytuju v subore /home/OS/predtest/vstupy-2/timem11.txt Cislo je suvisla postupnost cislic.Q

**tr -cs "[:digit:]" "\n" < /home/OS/predtest/vstupy-2/timem11.txt | grep -v '^$' | tr "\n" "+" | rev | cut -c2- | rev |cut bc**

mozete to spravit prikazom echo, do ktoreho si poslete vsetky cisla co sa nachadzaju v danom subore (poslat napr. subshellom) a potom treba pouzit prikaz bc

======================================================================

PRIKLAD 6

Zistite, kolko pouzivatelov ma nastavenu prihlasovaciu skupinu os2016 Uvazujte vsetkych pouzivatelov, ktori maju na stroji konto.

**cut -d ":" -f4 /etc/passwd | grep -c `grep "^os2016" /etc/group | cut -d ":" -f3`**

alebo

**grep -c `grep os2016 /etc/group | cut -d ":" -f3` /etc/passwd**

v /etc/group mate napisane, ktorej skupine ake cislo patri, to si vytiahneme a podla tohto cisla skupiny najdeme v /etc/passwd pouzivatelov, ktory maju pridelene toto cislo skupiny (subshell)

======================================================================

PRIKLAD 7

Zistite, ktory pouzivatel sa hlasi na osu z najvacsieho mnozstva roznych strojov a uvedte pocet tychto strojov.

Uvazujte kazde prihlasenie samostatne. Pouzite prikaz last a ako zdroj udajov pouzite subor

/home/OS/predtest/vstupy-2/wtmp.04Apr Ignorujte prihlasenia, pre ktore nepoznate adresu stroja.

**last -f /home/OS/predtest/vstupy-2/wtmp.04Apr | grep -v "^ wtmp" | cut -c 1-17,27-43 | sort | uniq | cut -d " " -f1 | uniq -c | sort | tail -n 1**

**last -f /home/OS/predtest/vstupy-2/wtmp.04Apr | cut -c1-17,23-43 | sort | uniq | cut -d " " -f1 | uniq -c | sort | tail -n 1**

uloha je myslim celkom jasna, jedine na co si dajte pozor su chybajuce stlpce - tych sa treba zbavit, taktiez aj vsetkych ,,skaredych'' riadkov, ktore tam nechceme

**PR3 Zistite, kolko pouzivatelov stroja osa ma meno Martin. Uvazujte vsetkych pouzivtelov, ktori maju na stroji konto.**

cut -d ":" -f5 /etc/passwd | grep "\<Martin\>" | wc -l

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_TEST\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PR1 Kolko netextovych obycajnych suborov je v adresari /home/OS/test/vstupy/test5**

file `find /home/... -type f` | grep -c -v 'text$'

find adresa -type f -exec file {} \; | grep -c -v 'text$'

**PR2[nedok] Zistite, na kolkych riadkoch v subore /home/OS/test/vstupy/timem11.tt koncia aspon tri vety. Za koniec vety povazujte slovo nasledovane niektorym zo znakov '.!?'.**

grep -c " [\.\?\!]\+3" /home/OS/test/vstupy/timem11.txt

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

grep -c -e "\(\>[\.\!\?].\*\)\{3\}" timem11.txt

{3}      -3 a viac krat, uzatvorkujeme aby to bolo skupina

**PR5[nedokoncene.. hviezdicka je problem] Zistite, kolko studentov malo najviac 20 bodov z cviceni a urobilo skusku. Pouzite udaje zo suboru /home/OS/test/vstupy/skuska2004.out**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

cut -c tx-xx,xx-xx < subor | grep -c "N" | cut -d " " -f1 | grep '\(^[0-9]$\)\|\(^1[0-9]$)\| ^20$'

**CV1**

/home/OS/ucebnove/01-04/os\_2-priklady

zacinaju s prvym riadkom:

#! /bin/csh

./1.csh -> spustenie skriptu..

#! - povie jadru ze ta cesta za tym bude ten program

csh -x <subor>     -ukaze ako sa to tam vykonavalo (debugger)

# - komentar

**cat 2.csh**

foreach u (steinmul lesko paulen)

        who | grep -w $u

end

u - iterator (kazda vec je slovo), iterovanie cez 3 prvky

\end      = koniec riadku

$u     - dolar je specialny znak, - hovori to co ide za nim je nazov premennej

namiesto dolar u tam hodi text.. steinmul

**cat 3.csh**

set veta = "Ahoj svet"

#set veta = Ahoj svet

@ pocet = 3

while ($pocet > 0)

        echo $pocet $veta

        echo $pocet     $veta

        echo "$pocet     $veta"

        echo '$pocet     $veta'

        echo \$pocet     \$veta

        echo ""

        @ pocet --

end

3 Ahoj svet

3 Ahoj svet

3     Ahoj svet

$pocet     $veta

$pocet $veta

2 Ahoj svet

2 Ahoj svet

2     Ahoj svet

$pocet     $veta

$pocet $veta

1 Ahoj svet

1 Ahoj svet

1     Ahoj svet

$pocet     $veta

$pocet $veta

set - prikaz, ktory do premennej priradi hodnotu

@ - prikaz, ktory robi matematicke operacie (premenna defaultne robi vzdy ako s textom),

set - samotny vypise vsetky premenne

apostrofy - zrusia aj dolare

echo - funguje tak, ze ked je tam viac medzier, zobere to ako jednu medzeru

set veta = ahoj svet      (spravi):

     set veta = ahoj

     set svet

set veta = Ahoj\ svet     (spravi):

     veta ="Ahoj svet"

*!     -substitucia*

*!his     - history*

*meno premennej - 2. substit.*

*alias - 3. subst.*

*\*.txt -     4. substitucia nazvu suboru*

*vystup priklazu      - 5. subs.*

**cat 4.csh**

foreach f ([A-Z]\*)

        echo "$f" | tr A-Z a-z

        mv -i "$f" "`echo '$f' | tr A-Z a-z`"

end

[A-Z]\*      - subory zacinaju velkym pismenom

pismena ktore zacinaju na velke pismeno, budu zmenene na male pismena

subshell - `` - medzi tym sa vytvaraju prikazy, a potom tam hodi vystup namiesto toho

`echo '$j'ls

"'$f'"

**CV2**

**cat 5.csh**

shift - posunie dolava pole (je destruktivny)

breaksw - vyskocenie zo switchu

-- -> koniec argumentu[prepinac]

polia su od jednotky v cshelli

unset debug

while ($#argv > 0)

        switch ("$argv[1]")

                case -d:

                        set debug

                        breaksw

                case -h:

                        echo "Help: ..."

                        exit 0

                case -\*:

                        echo "Neznamy prepinac $argv[1]"

                        exit 1

                default:

                        breaksw; break

        endsw

        shift

end

if ($?debug) echo "Dalsie argumenty: $argv"

**cat 6.csh**

:q       -> kazdy prvok pola ouvodzovkuje

set zoznam = (jeden dva tri styri pat "sest cele sedem")

echo $zoznam[2]

echo $#zoznam

echo ""

#set zoznam = $zoznam osem

#set zoznam = ($zoznam osem)

#set zoznam = ("$zoznam" osem)     -pridalo, ale aj sest cele sedem zobralo sa dalsie prvy, pocet prvkov je 9, co je nespravne

set zoznam = ($zoznam:q osem)     -ked si chceme pridat prvok do zoznamu, tak musime vytvorit na novo (:q -> kazdy prvok pola samostatne ouvodzovkuje)

#set zoznam = ($zoznam[1-6]:q sedem $zoznam[7-$#zoznam]:q)

set zoznam = ($zoznam[-6]:q sedem $zoznam[7-]:q)     -ulozi vsetko od 1 do 6 prvkov, siedmy bude sedem, a potom vsetko od 7. prvku sa tam ulozia zvysne ktore su od siedmeho.. cize sa tam vsunie ta sedmicka

set zoznam2 = ($zoznam:q)

while ($#zoznam > 0)          ->toto je rychlejsie ako to dolne

        echo "$zoznam[1]"

        shift zoznam

end

echo ""

@ i = $#zoznam2     -do icka sa da velkost pola

while ($i > 0)

        echo "$zoznam2[$i]"

        @ i --

end

echo ""

foreach z ($zoznam2:q)

        echo "$z"

end

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dva

6

jeden

dva

tri

styri

pat

sest cele sedem

sedem

osem

osem

sedem

sest cele sedem

pat

styri

tri

dva

jeden

jeden

dva

tri

styri

pat

sest cele sedem

sedem

osem

**cat 7.csh**

variable modifiers ... Cshell\

set cesta = /home/osop/prochazk/out.txt

echo $cesta:h     -vsetko po posledne lomitko nie vratane

echo $cesta:t     -posledny subor

echo $cesta:r     -vsetko pred bodkou

echo $cesta:e     -vsetko za poslednou bodkou

echo $cesta:r:t   -vsetko pred bodkou a zaroven posledny subor

/home/osop/prochazk

out.txt

/home/osop/prochazk/out

txt

out

**cat 8.csh**

ls -l

echo ""

foreach v (`ls -l | head -3`)

        echo "$v"

end

echo ""

foreach v ("`ls -l | head -3`")

        echo "$v"

end

[DLHY VYSTUP]

#! /bin/csh/hash)

ls

chmod u+x subor

./subor

csh subor

csh <subor