Operációs rendszerek alapjai

5. gyakorlat

Mai gyakorlat

- Szövegkezelés
- Feladatokhoz szükséges: G05.tar.gz

grep

- Globally search for a regular expression and print matching lines
- Több regex-t tud keresni több fájlban is, soronként
- Bemenet lehet pipe-olva
- `grep [OPTION...] PATTERNS [FILE...]`
  - `-c` kapcsolóval találatok számát írja ki
  - `-n` kapcsolóval sorok számának kiírása is
  - `-r` kapcsolóval rekurzívan keres
    - `-R` kapcsolóval szimbolikus linkeket is követ
- Példa
  - csak fájlok listázása: `ls -l | grep ^-`
  - `myfunc` függvény használatainak száma .cpp fájlokban: `grep -c myfunc *cpp`

G05F01

- Nézd meg, hogy milyen csomagok vannak feltelepítve
  - `dpkg` - Debian package
- Szűrj rá mondjuk a python szót tartalmazó csomagokra!
  - `dpkg -l | grep python`
  - `dpkg --get-selection | grep python`

G05F02

- Terminálban a nyilak használatával vissza lehet nézni a lefuttatott parancsokat
- `history` paranccsal kiírhatjuk az összeset (vagyis az utolsó n darabot)
- Szűrj rá például a `cd` parancsra!

grep

- A találat értelmezéséhez gyakran szükség van kontextusra
- Jó lenne, ha a találat körüli néhány sort is kiírná
  - például az előzmények szűrésekor
- `-A`, `-B`, `-C` kapcsolókkal találat előtt, után, előtt és után kiír n sort
  - `history | grep -C5 make`

sort

- Fájl sorait rendezi standard kimenetre
  - kimenet átirányítható
  - bemenet lehet pipe-olva
- Not-in-place rendez, önmagába irányított rendezés nem működik
- Néhány fontos kapcsoló
  - `-r`, `--reverse` fordítva rendez
  - `-R`, `--random-sort` random rendez
  - `-c` ellenőrzi a rendezést
  - `-t` elválasztó megadása
  - `-k` hanyadik oszlop szerint rendezzen

G05F03

- A points.txt fájlban a kedvenc tárgyad házi eredményeit láthatod .csv formátumban
  - NEPTUN,HF1,HF2
- A grades.txt fájlban az érdemjegyeket láthatod .csv formátumban
  - JEGY,NEPTUN
- Rendezd a points.txt fájlt Neptun kód szerint!
  - `sort points.txt`
- Rendezd a grades.txt fájlt Neptun kód szerint!
  - `sort -k 2 -t , grades.txt`

join

- Join lines of two files on a common field
- Rendezett fájlokra működik
- `join file1 file2`
- Néhány fontos kapcsoló
  - `-t` elválasztó megadása
  - `-1` első fájlban hanyadik oszlop szerint
  - `-2` második fájlban hanyadik oszlop szerint
  - `-o` kimeneti formátum megadása, például `1.1,1.2,2.1,2.2`

G05F04

- Fűzd össze a points.txt és grades.txt fájl tartalmát!
  - ne felejtsd el rendezni mindkét fájlt
- `join -2 2 -t , <(sort points.txt) <(sort -t , -k 2 grades.txt)`
- `join -22 -t, <(sort points.txt) <(sort -t, -k2 grades.txt)`
- Mentsd el az eredményt a class.txt fájlba!

cut

- Remove sections from each line of files
- Bemenet lehet pipe-olva is, kimenet standard outputra érkezik
- Fontosabb kapcsolók
  - `-b`, `-c`, `-f` kapcsolóval byte, karakter, vagy mezők kiválasztása
  - `-d` kapcsolóval elválasztó megadása (alapértelmezetten tab)
  - `--complement` kapcsolóval a komplemens kiválasztása
- Objektumok kiválasztása
  - N-edik objektum (egytől indexel): `N`
  - N-edik objektumtól: `N-`
  - N-edik objektumtól az M-edik objektumig (inkluzív): `N-M`
  - M-edik objektumig: `-M`
  - vesszővel elválasztva lehet felsorolni is

G05F05

- G05F04 feladat után class.txt fájlban megtalálhatok a pontszámok és a jegyek is
- Jelenítsd meg csak a Neptun kódokat!
  - `cut -d, -f1 class.txt`
  - `cut -c-6 class.txt`
  - `cut -b-6 class.txt`
- Jelenítsd meg csak a pontokat és a jegyeket!
  - `cut -d, -f2- class.txt`
  - `cut -d, -f1 --complement class.txt`

Egyéb szöveges parancsok

- `uniq`
  - ismételt sorokat keres, szűr
  - `uniq input.txt output.txt`
- `paste`
  - merge lines of files
  - `paste 1.txt 2.txt`
- `tr`
  - translate or delete characters
  - `tr [a-z] [A-Z] input.txt > output.txt`
  - `tr áÁéÉíÍóÓöÖőŐúÚüÜűŰ aAeEiIoOoOoOuUuUuU < input.txt`
- `shuf`
  - random permutációt ír ki (gyorsabb, mint a random sort)

comm

- Compare two sorted files line by line
- `comm input1.txt input2.txt`
- Kimenet három oszlopban
  - első oszlop: csak az első fájlban szereplő sor
  - második oszlop: csak a második fájlban szereplő sor
  - harmadik oszlop: mindkettő fájlban szereplő sor
- Fontosabb kapcsolók
  - `-1`, `-2`, `-3` kapcsolóval az első, második, harmadik oszlop mellőzése
  - `-i` kapcsolóval case-sensitivity kikapcsolható
  - `--output-delimiter` kapcsolóval állítható az oszlopok közti elválasztó

diff

- Compare files line by line
- Nem csak a fájlok közötti különbséget írja ki, azt is hogy milyen változtatás kell az egyezéshez
  - módtól függ, hogy hogyan írja ki
- Vegyük elő a múlt heti C++ példát!
  - old nevű mappában Hello World! main.cpp
  - new nevű mappában külön .hpp-ben és .cpp-ben Hello World! függvény
  - brief mód: `diff old new -q`

```
Only in new: func.cpp
Only in new: func.hpp
Files old/main.cpp and new/main.cpp differ
```

diff - normal mód (alapértelmezett)

```
vaghy@HAL9000:~/Desktop/G04$ diff old new
Only in new: func.cpp
Only in new: func.hpp
diff old/main.cpp new/main.cpp
1c1
< #include <iostream>
---
> #include "func.hpp"
4c4
<       std::cout << "Hello World!\n";
---
>       func();
```

- Csak a new mappában van a func.hpp és func.cpp fájl
- main.cpp fájlok különböznek
  - 1c1 - első fájl első sorát a második fájl első sorára kell cserélni
    - a, c, d: add, change, delete
  - < - első fájl adott sora
  - > - második fájl adott sora

diff - context mód (-c kapcsoló)

```
vaghy@HAL9000:~/Desktop/G04$ diff old new -c
Only in new: func.cpp
Only in new: func.hpp
diff -c old/main.cpp new/main.cpp
*** old/main.cpp        2020-10-05 15:18:01.312440761 +0200
--- new/main.cpp        2020-10-05 15:25:48.640721726 +0200
***************
*** 1,6 ****
! #include <iostream>

  int main() {
!       std::cout << "Hello World!\n";
        return 0;
  }
--- 1,6 ----
! #include "func.hpp"

  int main() {
!       func();
        return 0;
  }
```

- Első fájl * jellel jelölve, második - jellel
- Mindkét fájlt az elsőtől a hatodik sorig írja ki
- ! jelzi a különböző sorokat

diff - unified mód (-u kapcsoló)

```
vaghy@HAL9000:~/Desktop/G04$ diff old new -u
Only in new: func.cpp
Only in new: func.hpp
diff -u old/main.cpp new/main.cpp
--- old/main.cpp        2020-10-05 15:18:01.312440761 +0200
+++ new/main.cpp        2020-10-05 15:25:48.640721726 +0200
@@ -1,6 +1,6 @@
-#include <iostream>
+#include "func.hpp"

 int main() {
-       std::cout << "Hello World!\n";
+       func();
        return 0;
 }
```

- Első fájl - jellel jelölve, második + jellel
- Mindkét fájlt az elsőtől a hatodik sorig írja ki
- - jelzi a törlendő sorokat, + a hozzáadandó sorokat

diff

- Néhány fontosabb kapcsoló
  - `-y` kapcsolóval eredmény mutatása egymás mellett (csak normal módban)
  - `-r` kapcsolóval rekurzív (szimbolikus linkeket is követ)
  - `-x` kapcsolóval kihagyhatunk fájlokat vagy mappákat, amik megfelelnek a megadott mintának
    - `diff old new -x *.hpp`
  - `-X` kapcsolóval fájlban megadhatjuk a kihagyandó fájlokat
- Fájlba irányíthatjuk a kimenetet

patch

- Apply a diff file to an original
- A diff parancs unified kimenetében megadott utasításokat végrehajtja
  - lehet pipe-olni, argumentumként átadni vagy átirányítani
- Néhány fontosabb kapcsoló
  - `-b` kapcsolóval backup fájlok létrehozása (.orig végződés)
  - `-d` kapcsolóval belép egy mappába
  - `--dry-run` kapcsolóval végrehajtás nélkül kiírja, hogy mit csinálna
  - `-i` kapcsolóval argumentumként megadhatjuk a patchfile-t
  - `-R` kapcsolóval visszacsinálja

G05F06

- Hasonlítsd össze az old és a new mappa tartalmát!
  - `diff old new`
- Próbáld ki több módban is!
- Hajtsd végre az előírt változásokat! Készíts backup-t is!
  - `diff old new -u | patch -d old -b`
  - `diff old new -u > project.patch`
  - `patch -d old -b -i ../project.patch`
  - `patch -d old -b < project.patch`

sed

- Stream editor for filtering and transforming text
- Bemenet lehet fájl, vagy pipe-olva
- Alapértelmezetten kiír minden sort standard outputra
  - `-n` kapcsolóval kikapcsolhatjuk
- Legalább egy parancsot meg kell adni
  - üres parancs: `sed '' input.txt`
- `-e` kapcsolóval megadhatunk több parancsot is
  - `sed -e '' -e '' input.txt`
- `-i` kapcsolóval in-place végzi el a parancsokat
  - `-i.bak` kapcsolóval in-place dolgozik és az eredetiből backup-t csinál .bak kiterjesztéssel

sed

- Specifikálhatjuk, hogy az adott parancsot mely sorokra hajtsa végre
  - sorok számozása `1`-től `$`-ig
  - `n,m`: n-edik sortól m-edik sorig
  - `n~m`: n-edik sortól minden m-edik sor (itt használható a nulla is)
  - `range!`: megadott range negálása
  - `/regex/`: reguláris kifejezésnek megfelelő sorokra
  - `/start_regex/,/stop_regex/` : első regexnek megfelelő sortól a második regexnek megfelelő sorig
  - lehet kombinálni is a számozás és a reguláris kifejezéseket
- Több parancs végrehajtása sorok kiválasztása után
  - `-e` kapcsolóval fapados, mert többször be kell írnom a szűrőt
  - `{}` között felsorolt parancsokat végrehajtja a szűrés után
    - új sor az elválasztó

sed

- `p` - print
- `d` - delete
- `a` - append
  - sor után beilleszt új sorba
  - `sed 'a\new line after each line' input.txt`
- `c` - change
  - megváltoztatja a sort
  - `sed '0~2 c\changed every second line' input.txt`
- `i` - insert
  - sor előtt beilleszt új sorba
  - `sed '/regex/ i\new line before lines that matched' input.txt`

sed

- `s` - substitute
  - `sed 's/old_regex/new_regex/' input.txt`
  - tetszőleges elválasztót használhatunk (ez akkor jó, ha a regex tartalmazza a / karaktert)
  - parancs végére írhatjuk a beállításokat
    - `'s/old_regex/new_regex/g'` összes találatot módosítja
    - `'s/old_regex/new_regex/2'` csak a második találatot módosítja
    - `'s/old_regex/new_regex/5g'` ötödik találattól kezdve összeset módosítja
- `r` - read
  - lehet megadott fájlból olvasni
  - szokásosan lehet kezelni a sorokat
- `w` - write
  - lehet megadott fájlba írni

G05F07

- A fruits.txt fájlban láthatsz egy listát, hogy adott gyümölcsökből mennyi áll rendelkezésre
- Jelezd az adott sor végén, ha elfogyott az adott gyümölcs!
  - `sed '/ 0 / s/$/ elfogyott:(/' fruits.txt`
- Próbáld ki in-place és csinálj backup-t is!
  - `sed -i.bak '/ 0 / s/$/ elfogyott:(/' fruits.txt`
- Töröld ki a sort és írd a helyére az üzenetet!
  - `sed '/ 0 / c\elfogyott:(' fruits.txt`
- Próbáld fájlból beolvasni a hiba üzenetet!
  - ```
    sed '/ 0 / {r fruits_error.txt
    d}' fruits.txt
    ```
    ```

sed

- Van lehetőségünk létrehozni egy sed script-t is
  - érdemes, ha van egy nagyon jól megírt eljárásunk
- Minden parancs új sorban
- `sed -f script.sed input.txt`
- Futtathatunk interpreter direktívával is
  - `#!/bin/sed -f`

G05F08

- Írd ki a Neptun kódokat és a jegyeket a class.txt fájlból!
- Írj egy sed script-t, ami a jegy alapján külön fájlokba teszi a hallgatókat!
  - bukó hallgatók
  - elégséges és közepes hallgatók
  - jó és kitűnő hallgatók
- Írj egy script-t, ami a megfelelő template levelek alapján leveleket ír!
  - levél neve legyen a Neptun kód
  - hallgató Neptun kódját a <NEPTUN> kulcsszó helyére kell írni
  - hallgató jegyét a <GRADE> kulcsszó helyére kell írni

Következő gyakorlat

- awk

Érdekességek, olvasnivalók

- Brian Kernighan interviews Ken Thompson
- Brian Kernighan: Where GREP Came From
- Myers' Difference Algorithm
- Nagyon jó sed tutorial