5. Shell script I.

Lépjünk be terminálkapcsolattal a kiszolgálóra (**szamrend.inf.elte.hu**).

Készítsünk egy ideiglenes könyvtárat, amibe elhelyezhetjük az órán készített állományainkat (pl. mkdir szamrend5; cd szamrend5).

1. Shell script parancsfájlok:
    a. Parancsfájl készítés
        i. Szöveges fájl, x jog, chmod parancs
        ii. # megjegyzés
        iii. egy sorba több parancs is írható, a ; az elválasztó
        iv. echo almafa; echo alatt # ez az első program (echo –n)
    b. Parancsfájl futtatás
        i. ./parancsnév
        ii. bash parancsnév
        iii. Script első sora gyakran ilyen: #!/bin/sh
            1. Jelentés: Ki értelmezze a fájlt.
    c. .profile állomány
        i. Belépéskor lefut. Alapból ezen a gépen nincs.
        ii. Feladat: Készíts magadnak egy .profile állományt, ami üdvözöl belépéskor!

2. Változók, paraméterek, beolvasás (read):
    a. Környezeti változók, env parancs
        i. PS1 változó, prompt értéke, bash-ban, \u= usernév, \h=hostnév
            1. PS2, másodlagos prompt
        ii. Egy változó értéke: $név
        iii. PATH változó
            1. Feladat: Bővítsük a PATH változót a . könyvtárral! (*PATH=$PATH:.*) Miért a .profile állományba írjuk a módosítást?
    b. Összes változó, set parancs
        i. Saját változó definiálás: *n=alma*
        ii. Shellben látjuk, de scriptben nem. *export n* , ettől n környezeti változó lesz, látszik scriptben.
        iii. Változó megszüntetése: *unset n*
    c. Script paraméterek
        i. $0 parancs neve, $1 első, $2, …$9 kilencedik paraméter
            1. Feladat: állapítsuk meg, mi a shell program neve, amiben éppen dolgozunk, illetve milyen paraméterekkel van indítva.
        ii. $# paraméterek száma
        iii. $* összes paraméter
    d. Feladat: Írjon shell parancsot ami kiíja a paramétereinek számát és az összes paramétert!
    e. Feladat: Írjon shell parancsot, ami a paraméter értékénél eggyel nagyobbat ad meg! Írja ki a paraméter kétszeresét is! *expr*
    f. Feladat: Írjunk shell parancsot, ami bekéri a hallgató nevét, és életkorát, majd megjeleníti azt!

3. Szűrők, csövek (pipe):
    a. Parancs kimenet egy másik parancs bemeneteként jelenik meg: | jel
        i. more, head, tail, sort parancs: cat nagyfile|more
            i. Feladat: Írjunk parancsot, ami rendezve kiírja egy fájl első és utolsó n sorát. n legyen paraméter!
        ii. wc parancs, példa: cat .profile|wc
            i. Feladat: mondjuk meg, hány bejegyzés van a könyvtárunkban!
        iii. cut parancs
            i. Feladat: Írjunk shell programot, ami az első és utolsó paramétert kiírja, majd összeadja őket! (utolsó=`echo $*|cut –f$# -d” ”`)
            ii. Feladat Kérdezzük le a rendszer összes felhasználójának adatait a `getent passwd` parancs segítségével és tegyük bele *felhasznalok.txt* állományba, hány felhasználóról kapunk információt?
        iv. grep parancs
            i. Feladat: Határozzuk meg, hogy hány z betűvel kezdődő felhasználó van bejelentkezve! (who|grep ^i|wc -l)
            ii. Feladat: Írjuk bele a *torolt_felhasznalok.txt* állományba a rendszerből ideiglenesen törölt felhasználók nevét! Segítség: az ideiglenesen törölt felhasználók a 666-os felhasználó-azonosítót kapják. (`grep ':666:' < felhasznalok.txt | cut -f1 -d: > torolt_felhasznalok.txt`)

4. Input-Output átirányítás
    a. > output átirányítás
        i. Feladat: Mit jelent: echo alma >&2 ? (Kimenet, a hibakimenetre megy.)
        ii. Feladat: Irányítsuk a kimenetet a ’feneketlen kukába’! (echo kukába >/dev/null)
    b. >> hozzáfűzés
    c. < input átirányítás
    d. << here input
        i. Feladat: Mit jelent?

        ```
        cat <<alma
        <title> Shell script 1. </title>
        <body>  Programjaim: </body>
        </html>
        alma
        ```

        (cat paramétere lesz a cat <<alma, alma közti sorok)
    e. 2> hiba kimenet átirányítás („2>&1” a hiba kimenet és az egyszerű kimenet összefűzése, kukába a hibával: 2>/dev/null)

5. Töröljük az ideiglenes állományokat, lépjünk ki a terminálprogramból, majd az operációs rendszerből.

6. Ha időnk és kedvünk engedi...

a. Színekkel is felturbózhatjuk a promptot, pl. az `export PS1="[\033[01;32m\]\u@\h\[\033[01;34m\] \w \$\[\033[00m\]"` kiadását követően zöld és kék színben jelennek meg a számunkra fontos információk. Az ANSI kódok táblázatát a http://en.wikipedia.org/wiki/ANSI_escape_code lapon találjuk.
b. Módosítsuk a .profile-t úgy, hogy belépés után az előbbi prompt jelenjen meg.
c. Feladat: Mit csinál a `cat <a.txt >>a.txt` parancs (létező a.txt esetén) és miért rossz ez nekünk? A feladat (megoldást, kipróbálást nem feltétlenül kell megcsinálni) megbeszélése során hívjuk fel a hallgatók figyelmét a *CTRL-C* kombinációra, illetve a befejezést követően ha kipróbáltuk, mindenképp töröljük az a.txt-t!

7. Segédanyag(ok):
   a. bash tutorial: http://www.freeos.com/guides/lsst/
   b. bash szkriptek gyűjteménye: http://bash.cyberciti.biz/