2. Gyakorlat

Operációs rendszer:

- Programok végrehajtásának ütemezéséért, erőforrások menedzseléséért, felhasználó és számítógép közötti kommunikációért felel.
- Alapvetően három részre bontható: felhasználó felület (shell), alacsony szintű segédprogramok, kernel.
- Feladata:
  - Mint kiterjesztett gép: magasabb absztrakciós szintet biztosít a felhasználó számára, elfedi az eszközök közötti különbségeket.
  - Mint erőforrás menedzser: elosztja az erőforrásokat a programok között, a menedzselési feladatkörbe az erőforrások kiosztása mellett azok védelmét is beleértjük.

Kernel:

- Több különböző típus:
  - Monolitikus-kernel: nincs strukturáltság, a rendszer magja névvel ellátott szolgáltató eljárások gyűjteménye, szolgáltató rutinok hívhatók explicit (system call) és implicit (exception, interrupt) módon.
  - Hybrid-kernel,
  - Mikro-kernel: kliens-szerver modell,
  - Exo-kernel.
- Feladata: IO eszközök, programok futásának, háttértárolók kezelése, rendszerhívások kiszolgálása, fájlrendszer biztosítása, stb.

Shell:

- Feladata: kapcsolattartás a felhasználóval, a programok futásának kezelése.
- UNIX rendszerhéj:
  - UNIX alapú rendszerekben különböző parancsértelmezők működnek, ezeket rendszerhéjnak (shell) nevezzük.
  - A felhasználó több rendszerhéj közül választhat, pl.: sh, bash, ksh, csh, stb.
  - A rendszerhéj shellscript-ek segítségével programozható.

Parancs:

- Fehér karakterekkel (whitespace) határolt szavak sora.
- Az első szó a parancs neve, a többi a hozzá tartozó argumentumok sora.
- A parancsokat a shell beolvassa, értelmezi, átalakítja, végrehajtja.
- Belső parancs: a shell maga hajtja végre, pl.: cd, export, umask, stb.
- Külső parancs: olyan parancsok, melyekhez tartozik valamilyen bináris program, melynek futása a shell egy gyerekprocesszében történik.

Processz:

- A processz egy végrehajtási példánya egy párhuzamosságot nem tartalmazó végrehajtható programnak.
- A processz egy futó program példány.
- A klasszikus processz egy szálon fut, a benne futó végrehajtható program nem tartalmaz párhuzamosságot.

Taszk, fonál:

- A modern OS-ek támogatják a párhuzamosságot, így a processz helyett két másik fogalom jelenik meg: taszk (task), fonál (szál, thread).
- A futó program egy taszk, amely több fonalat is tartalmazhat, melyek egymással is versengenek a CPU-ért.
- Egy taszk egyetlen fonállal a klasszikus processz.

UNIX, Linux sajátosságok:

- Multiuser, multitasking,
- Minden fájl (egyszerű fájlok, jegyzékek, eszközök, stb),
- Case-sensitive, legfeljebb 255 karakter hosszú fájl nevek,
- A fájlnév nem kezdődhet . karakterrel (speciális szerepe van), és nem tartalmazhat * karaktert.

Jogosultságok:

- A hozzáférési jogok (r, w, x) egyszerű ábrázolásához 3 számjegyből álló kódot használunk. Pl.: 400, 644, 755, 777, stb.
- Az első számjegy a fájl tulajdonosának, a második a felhasználó csoportjainak, az utolsó pedig mindenki másnak a hozzáférési jogait jelöli.
- Egyszerű fájlok esetén: r = olvasható, w = írható, x = futtatható.
- Jegyzék esetén: r = listázható, w = tartalma módosítható, x = bele lehet lépni (cd).
- Jogosultságok beállítása: chmod (csak a fájl tulajdonosának van rá joga).
- Fájl létrehozásakor használt maszk megadása: umask (azokat a jogokat kell beírni, amiket nem akarunk megadni).
- Tulajdonjog átadása: chown, chgrp (csak a fájl tulajdonosa teheti meg).
- SUID bit:
  - SUID = Set User Identification.
  - Alapesetben a programok az azt futtatni akaró felhasználó jogkörével futnak, ha be van állítva a SUID bit, akkor a program tulajdonosának jogkörével.
  - Beállítása:
    - Szimbolikusan: chmod +s fájlnév
    - Numerikusan: chmod 4755 fájlnév
    - Eredménye pl.: -rwsr-xr-x 1 alex alexcsoportja 148 Feb 18 16:37 fájlnév
- Az összes SUID bites program megkeresése a rendszerben: find / -perm +4000
- SGID bit:
  - SUID bithez hasonló, csoportokra vonatkozik.
  - Beállítása: chmod 2755 fájlnév

Archiválás:

- Az adatveszteség elkerülésére hatékony megoldás a biztonsági mentések készítése, amit elvégezhetünk a **tar** (Tape ARchiver) program segítségével.
- A program használatának szintaktikája: tar [kapcsolók] [archívum] [fájlok]
- A program argumentumai a végrehajtandó művelettől függnek.
- A leggyakrabban használt kapcsolók:
  - **c**: create, archívum létrehozása,
  - **f**: file, átirányítás fájlba,

  - **x**: extract, archívum kicsomagolása,
  - **t**: test, archívum tartalmának listázása,
  - **u**: update,
  - **v**: verbose, beszédes kimenet,
  - **z**: zip, tömörítő eljárás használata (általában gzip, vagy bzip2).
- A tar-ral készített archívumok kiterjesztése általában: tar, tar.gz, vagy tgz.
- Példák:
  - tar czf ~/mentes.tar.gz ~/public_html
  - tar tf ~/mentes.tar.gz
  - tar xvzf ~/menter.tar.gz

Szinkronizálás:

- Fájlok két gép közötti szinkronizálására, valamint biztonsági mentések készítésére az **rsync** program használata ajánlott.
- A program használatának szintaktikája: rsync [kapcsolók] [forrás] [cél]
- Az argumentumok a végrehajtandó művelettől függnek, melyek a következők lehetnek:
  - **–a**: archive, a másolás során megmaradnak a szimbolikus linkek és a jogosultságok.
  - **–z**: compress, az adatok tömörítve másolódnak.
  - **–b**: backup, biztonsági mentés készítése.
  - **–u**: update, csak a régebbi fájlokat írja felül.
  - **--partial**: részlegesen letöltött fájlok nem törlődnek.
  - **--progress**: folyamatjelző.
  - **–P**: --partial és --progress együtt.
  - **--exclude**: fájlok kihagyása.
  - **–e**: remote shell használata.
  - **–v**: verbose, beszédes kimenet.
- Példák:
  - rsync fájlok felhasználónév@host:útvonal
  - rsync mentes.tar.gz toth130@jerry:~/dummy_directory
  - rsync –avzP local_dummy_directory jerry:~/remote_dummy_directory
  - rsync –e ssh –avuz --exclude '*~' --progress ~/local_dir jerry:~/remote_dir
    rsync –e ssh –avuz --exclude '*~' --progress jerry:~/remote_dir ~/local_dir