Operációs rendszerek - 1

I. Bevezetés

Az operációs rendszer fogalma

Olyan programrendszer, amely a számítógépes rendszerben a programok végrehajtását vezérli. Így például ütemezi a programok végrehajtását, elosztja az erőforrásokat, biztosítja a felhasználó és a számítógépes rendszer közötti kommunikációt.

Az operációs rendszerek helye

- Felhasználó
- Alkalmazás
- **Operációs rendszer**
- Hardver

Az operációs rendszer feladatai

- Kapcsolat teremtése a felhasználó és a gép között
- Biztosítja az adatok elérhetőségét
- Processzor vezérlése
- Programok működtetése: indítás, programok közötti kapcsolatok szervezése
- Háttértárak kezelése: programok, adatok biztonságos tárolása
- Perifériák kezelése: berendezések vizsgálata, az I/O igények sorba állítása
- A memória kezelése (lefoglalás, programok betöltése, memória felszabadítása, lapozás, virtuális tárkezelés)
- A gépi erőforrások elosztása (erőforrás pl.: háttértár, memória, hálózat, megjelenítő egység, nyomtató)
- Kommunikáció, kapcsolattartás a gép kezelőjével (parancsok fogadása, üzenetek küldése)

Az operációs rendszer fejlődése

- Egyszerű kötegelt rendszerek
- Multiprogramozott kötegelt rendszerek
- Időosztásos rendszerek
- Személyi számítógépek

Miért jobb a parancssoros megközelítés?

- Gyorsabb
- Áttekinthetőbb
- Optimalizáltabb
- Hatékonyabb

DOS (Disk Operating System)

- Parancssoros
- Egyfelhasználós
- Monoprogramozott

PowerShell

- Objektumorientált parancsfeldolgozó és szkriptkörnyezet
- A Microsoft fejlesztette ki, mert a DOS nem volt elég hatékony grafikus felületen

Unix

- Kezdetben nyílt forráskódú, általános célú operációs rendszer
- Többszálás, többfeladatos és többfelhasználós
- A rendszermag és a felhasználói felület elágazik egymástól

A többfeladatos és többfelhasználós operációs rendszerek sajátossága, hogy egyszerre több felhasználó is dolgozhat ugyanazon a gépen és az operációs rendszer egyszerre több feladatot is el tud látni.

Linux disztribúciók

Arch Linux	CentOS	Damn Small	Debian	DreamLinux	Fedora	Gentoo linux	gOS
Knoppix	Mandriva	Mepis	Novell	openSUSE	PCLinuxOS	Puppy	Redhat
Sabayon	Slackware	System V	Ubuntu	Vector	Xandros	Zenwalk	

Unix felépítése

- Alkalmazások
- Segédprogramok
- **Shell**
- Kernel
- Hardver

Shell

- Saját parancsok kivételezése
- Héjprogramok (szkriptek) végrehajtása
- Feltételes kifejezések alkalmazása
- Pszeudo parancsok létrehozása
- Belső változók használata
- Be és kimenet átirányítása
- Folyamatok indítása a háttérben

A program fogalma

A program a számítógépnek szóló utasítások sorozata, amely egy kidolgozott algoritmus alapján meghatározza, hogy a számítógép milyen módon végezzen el egy adott feladatot. Ha egy programot elindítunk, az operációs rendszer a háttértárolóról betölti a memóriába, a CPU számára átadja a program kezdetének címét, majd a program ezután átveszi a számítógép vezérlését és futni, működni kezd. A **folyamat** a program egy futó példánya, amely saját adatterülettel rendelkezik.

Démonok

Speciális háttérfolyamatok, melyeket a rendszer futtat, és valamilyen felügyeleti szerepet látnak el a háttérben.

II. Fájlkezelés

A Linux alapfilozófiája

A könyvtárakat katalógusoknak nevezzük, amelyeket – csak úgy mint minden mást - fájloknak tekintünk.

INODE (Index-node)

Minden UNIX fájlrendszer elején megtalálható az úgynevezett **szuperblokk**, amelyik a fájlrendszer legfontosabb adatait, a belső táblák és azonosítók méretét, stb. tartalmazza. Ezt követi az úgynevezett inode tábla, végül a ténylegesen felhasználható lemezterület, ahol az állományok által lefoglalt és a még szabad lemezblokkok vegyesen helyezkednek el.

Az inode név az index-node bevett rövidítése, és a fájlrendszer kialakításának fontos tényére utal, nevezetesen arra, hogy az állományok jellemzőit tartalmazó inode-okat indexként használják a rendszerprogramok. Az **inode tábla** egy fix méretű tábla, fix hosszúságú rekordokkal. Minden egyes fájlhoz egy és csakis egy inode bejegyzés tartozik. Az inode tartalmazza az adott fájlra vonatkozó összes lényeges információt, az állomány méretét, típusát, tulajdonosát és csoportját, a hozzáférési jogokat, és az állományt alkotó lemezblokkok fizikai elhelyezkedését a lemezen.

Az állomány nevét a **katalógus**(fájl) tartalmazza. A katalógusfájlban minden egyes fájlhoz, ami az adott katalógusban szerepel, egy bejegyzés tartozik. E rekord nem tartalmaz mást, mint az állomány inode számát, és az állomány nevét.

A fájlrendszer kitűntetett pontjai

- A fájlrendszer kezdőpontja (root)
- Az egyes felhasználókhoz tartozó kiindulási pont (home)
- Az aktuális katalógus

A **pwd** (print work directory) parancs hatására a képernyőre íródik az aktuális katalógus neve.

Rejtett fájlok

Azon állományok, amelyek neve ponttal kezdődik rejtett fájloknak tekintendőek. Ezek az állományok csak akkor jelennek meg listázáskor, ha azt explicit módon kérjük (ls –a). Ilyenek általában a konfigurációs állományok.

Merev láncolás (Hard link)

A merev láncolás legfőbb jellemzője, hogy a láncolt fájl teljesen egyenértékű az eredetivel, hiszen egyazon inode-ra mutat két egyenértékű katalógus bejegyzés, amelyek között nem lehet különbséget tenni.

Lágy, vagy szimbolikus láncolás (Soft link)

Lényege, hogy a szimbolikus link katalógus bejegyzése nem a fájl inode-jára mutat, hanem egy olyan különleges fájlra, ami a láncolt fájl nevét tartalmazza. Szimbolikus linket szintén ln paranccsal hozunk létre – úgy mint merev láncolásnál -, de a -s opciót is meg kell adni.

III. Jogosultságkezelés

Minden fájl és könyvtár rendelkezik hozzáférési jogokkal.

Az elérést 3 szinten korlátozhatjuk

- Owner (tulajdonos)
- Group (csoport)
- Others (többiek)

A hozzáférési jogokat a **chmod** (change mode) paranccsal módosíthatjuk.

3 különböző művelet értelmezett

- Olvasás (**r**ead)
- Írás (**w**rite)
- Futtatás (e**x**ecute)

A chmod parancs működése

\+ Jog hozzáadása
\- Jog elvétele

Melyik csoportot érintse a változtatás

- **U**sers
- **G**roup
- **O**thers
- **A**ll

Példa a használatra

chmod g+rw fajlnev.kit
chmod o+r-w fajlnev.kit

A chmod parancsnak két használati módja van, az új jogosultságokat oktális módon is be lehet állítani. Ekkor a jogokat 3 szám formájában adjuk meg úgy, hogy az első szám a **felhasználó** jogait, a második szám a **csoport** jogait, a harmadik szám pedig a **többiek** jogait írja le. A számok jelentése a táblázatból leolvasható.

Decimális	Bináris	Olvasás	Írás	Futtatás
0	000	0	0	0
1	001	0	0	1
2	010	0	1	0
3	011	0	1	1
4	100	1	0	0
5	101	1	0	1
6	110	1	1	0
7	111	1	1	1

Az **umask** (user mask) paranncsal lekérdezhető az alapértelmezett maszk és módosítható az. Alapértelmezett esetben bináris formában írja ki. Az –s opciót választva szimbolikus formában írja ki.

Lehetőségünk van arra is, hogy az általunk birtokolt állomány tulajdonosát és csoportját megváltoztassuk. Ez a **chown** (change own), illetve a **chgrp** (change group)paranccsal történik.

IV. Katalóguskezelés

Lépegetés a katalógusok között

A katalógusok között a **cd** (change directory) parancs segítségével mozoghatunk. Ha semmilyen értéket nem adunk meg neki, akkor a felhasználói főkönyvtárba (home) ugrunk vissza.

Katalógus létrehozása

Új katalógusokat az **mkdir** (make directory) paranccsal hozhatunk létre. Egyszerre több katalógusnevet is megadhatunk, a rendszer mindegyiket létrehozza. Ha a létrehozandó katalógus egy olyan katalógus alatt helyezkedne el, amely maga sem létezik, akkor a katalógus létrehozása sikertelen lesz. Ebben az esetben a -p opcióval érdemes kiadni a mkdir parancsot; ennek hatására a az összes szükséges szülő katalógust létrehozza a rendszer.

Katalógus törlése

Már létező katalógust az **rmdir** (remove directory) paranccsal törölhetünk. (Most is megadhatunk több paramétert.) Fontos megjegyezni, hogy e paranccsal csak üres katalógust törölhetünk, ha a katalógusban fájlok vannak, a UNIX nem hagyja a katalógus törlését, előbb törölni kell a fájlokat. A –p (posix) opciót megadva az almappákat is törli, amennyiben azok üresek.

Katalógus tartalmának listázása

A katalógusok tartalmát az **ls** (list) paranccsal listázhatjuk ki. Ha semmi mást nem mondunk, az aktuális katalógusban lévő fájlok nevét listázza ki, ábécésorrendben.

ls –i	A fájl indexének számát is megjeleníti. (inode)
ls –a	Minden fájlt - a rejtetteket is - kilistáz. (all)
ls –R	Minden könyvtár tartalmát rekurzívan kilistázza. (recursive)
ls –F	A fájlnevekhez egy, a fájl típusát jelző karaktert fűz. Szabályos végrehajtható fájl esetén ez egy `*'-jel, könyvtár esetén `/', szimbolikus kötés esetén `@', FIFO esetén `|', socketek esetén `=', más esetekben semmi. (classify)
ls –l	Részletes információk listázása: jogok (10 karakter), linkek száma, felhasználói név, csoport, állomány mérete blokkokban, utolsó módosítás ideje, állomány neve. (long)

V. Fájlműveletek

Fájlok másolása és áthelyezése

A fájlok másolására a **cp** (copy) parancs szolgál. Fontos különbség a DOS-hoz szokott felhasználó számára, hogy itt mindenképp legalább két argumentumot meg kell adni, azaz a DOS alatt szokásos copy fájl megadásmód nem megy, a célfájlt vagy katalógust is meg kell adni. A cp parancs működése hasonló a **mv** parancshoz, azaz itt is fájlt fájlba másol, ha a céltárgy fájl, illetve katalógusba helyez, ha a céltárgy katalógus.

Fájlok törlése

Fájlokat az **rm** (remove) paranccsal törölhetünk, természetesen most is megadhatunk több fájlnevet a paraméterlistán. Az rm parancs opciói közül gyakran fontos lehet a –f. Ennek hatására az egyébként írásvédett fájlokat is törli az rm.

VI. Keresés

Find parancs

Fájlokat keres egy könyvtárstruktúrában.

find [útvonal…] [opció] ’ [kifejezés] ’

find –name	nev	A fájlnév alapja illeszkedik a nev burokmintához.
find –group	nev	A fájl a nev csoporthoz tartozik. (Numerikus érték is megengedett.)
find –size	n	A fájl mérete n egységnyi. Az egység az 512-bájtos blokk alapértelmezésben vagy `b' végződés esetén. Bájt = ’c’ Kilobájt = ’k’ 2-bájtos szó = ’w’ végződés esetén.
find –mtime	n	A fájl adatai utoljára n*24 órája lettek módosítva.
find –newer	fajl	A fájl frissebben lett módosítva, mint fajl.
find –type	c	A fájl c típusú, ahol a lehetséges értékek: - b blokkos (pufferelt) speciális eszközfájl (block) - c karakteres (nem pufferelt) speciális eszközfájl (character) - d könyvtár (directory) - p csőhálózat (pipe) - f szabályos fájl (file) - l szimbolikus kötés - s socket

VII. Átirányítás

A standard átirányítás mechanizmusának a UNIX alatt kitüntetett jelentősége van: gyakorlatilag minden, a standard outputra író program kimenete átirányítható egy tetszőleges állományba, s hasonlóképp, bármelyik program, amelyik a standard inputról olvas, tetszőleges állományból veheti inputját. A bemenet átirányításának jele a **<** karakter, a kimeneté a **>** karakter. A **<<** és **>>** karakterpárosok hasonló célt szolgálnak, azonban ezek nem írják felül az állomány eredeti tartalmát, hanem hozzáfűznek ahhoz.

A **>&** karakterpáros a standard hibakimenetet irányítja át egy állományba. A **>!** Karakterpáros pedig csupán annyiban különbözik a **>** karaktertől, hogy a speciális noclobber shell változó beállításaitól függetlenül is végrehajtódik.

VIII. Csővezetékek

Gyakran előfordul, hogy egy program egy másik program kimenetét használta fel bemenetként, egy ideiglenes állomány közbeiktatásával. Ennek és az ehhez hasonló feladatoknak a megoldására szolgál a csővezeték (pipe). A pipe, melynek jele a | karakter, az egyik program kimenetét a másik program bemenetével köti össze. A pipe-ok többszöröshetőek is, azaz láncszerűen egymásra épülve adhatják tovább az adatokat egymásnak. A **|&** karakterpáros a kimenetet és a hibacsatornát irányítja át.

Csővezetékek elágaztatása

A csővezetékeket a **tee** parancs segítségével tudjuk elágaztatni. Bemenetét szintén a standard inputról veszi, de kimenetét két helyre küldi; egyrészt a standard outputra, másrészt a paraméterként megnevezett állományba. Ez elsősorban programbelövéseknél igen hasznos szolgáltatás. A **-a** opció hatására a paraméterként megadott kimeneti állományt nem írja felül, hanem hozzáfűzi az újabb kimeneti adatokat.

IX. Állományok összehasonlítása

Comm parancs

A **comm** három oszlopban írja ki két állomány összehasonlításának eredményét. Az első oszlopban csak azok a sorok szerepelnek, amelyek csak az első, a másodikban azok amelyek csak a második, végül a harmadikban azok amelyek mindkét állományban előfordulnak. Ha az egyik állománynév helyén **-** jelet írunk, akkor a standard inputról várja az egyik összehasonlítandó tartalmat. Így interaktívan használható arra is, hogy megnézzük bizonyos sorok meglétét egy állományban. Ha a comm paranccsal összehasonlítjuk semmit és valamit az eredmény csak egy közös sort fog mutatni, az első, csupa csillagból álló sort. Értelmesen csak akkor tud a comm dolgozni, ha az állományok sorai "ábécé"-be rendezettek.

Cmp parancs

A **cmp** parancs két állomány összehasonlítására szolgál, itt is szerepelhet az egyik állomány helyén **-** karakter, a standard inputot jelölve. Alapértelmezés szerint azonos állományok esetén nem ír ki semmit, eltérés esetén jelzi az eltérés sor- és bájtszámát. Felismeri, ha az egyik állomány a másik kezdeti része. Nagy előnye, hogy a diff paranccsal ellentétben bináris fájlok összehasonlítására is alkalmas.

Diff parancs

A **diff** parancs kilistázza a két összehasonlított állomány eltérő részeit, ----- karakterekkel jelölve az egyes állományokban eltérő részeket. A diff igazi használhatósága azonban abban rejlik, hogy nemcsak azt mondja meg, hogy a két állomány hol tér el egymástól, hanem azt is, hogy hogyan lehet az egyikből a másikat rekonstruálni.

X. Szűrőparancsok

Cat parancs

A **cat** (concatenate) parancs állományok összefűzésére és kilistázására szolgál. A paraméterként megnevezett fájlokat - ennek hiányában pedig a standard inputot, azaz a billentyűzetről, pipe-ból - érkező adatokat listázza ki, folyamatosan, tördelés nélkül.

cat –n	Minden kimeneti sort megszámoz 1-től kezdődően. (number)
cat –b	Minden nemüres kimeneti sort megszámoz 1-től kezdődően. (number-nonblank)
cat –s	Az egymás után ismétlődő üres sorokat egyetlen üres sorral helyettesíti. (squeeze-blank)

Head parancs

A **head** parancs a bemenet „első részét” írja a kimenetre. Alapértelmezetten az első 10 sor.

head –c N	Az első N bájtot írja ki. (char)
head –n N	Az első N sort írja ki. (line)

Tail parancs

A **tail** parancs a bemenet „utolsó részét” írja a kimenetre. Alapértelmezetten az utolsó 10 sor.

tail –c N	Az utolsó N bájtot írja ki. (char)
tail –n N	Az utolsó N sort írja ki. (line)

Wc parancs

A **wc** (word count) a bájtok, szavak és újsor-jelek számát számolja meg az argumentumként megadott fájlokban. Ha nem adunk meg fájlnevet, illetve a fájlnévként a ` - ' jelet adjuk meg, akkor a standard bemenet olvassa a program. Alapértelmezés szerint a wc mindhárom számot kiírja. Az opciókkal lehet megadni, hogy csak bizonyos számok legyenek kiírva. Az opciók nem semlegesítik egymás hatását, így pl. wc --bytes --words a bájtok és a szavak számát egyaránt kiírja. Minden fájlról egysornyi információt ír ki, és az argumentumként megadott fájlok nevét is kijelzi. A megadott adatok sorrendben a következőek: sorok, szavak, bájtok száma.

wc –c	Csak a bájtok számát írja ki. (chars)
wc –l	Csak a sorok számát írja ki. (lines)
wc –w	Csak a szavak számát írja ki. (words)
wc –L	Csak a fájlban előforduló leghosszabb sor hosszát írja ki. (max-line-length)

Uniq parancs

A **uniq** (unique) parancs kiírja az egyedi sorokat egy rendezett fájlból, és eldobja az egyezőket egy kivételével. Rendezett bemenetre van szüksége, mivel csak az egymás után következő sorokat hasonlítja össze.

uniq –c	Kiírja a sor elé, hogy az adott sor hányszor fordult elő. (count)
uniq –d	Csak a duplikált sorokat írja ki. (repeated)
uniq –i	Kis és nagybetűk figyelmen kívül hagyása. (ingore)
uniq –u	Csak a nem azonos sorokat írja ki. (unique)

Sort parancs

Az Állományok rendezésére a standard Unixban egyetlen, ámde igen erőteljes parancs áll rendelkezésre, a **sort**. Ha bemenet nincs megadva, a standard inputot olvassa és rendezi. A kimenet alapértelmezés szerint a standard output, a **-o** opcióval lehet egy fájlba irányítani a rendezés eredményét. Alapértelmezésben a sort parancs soronként, ASCII karakterérték szerint, növekvő sorrendben, a teljes beolvasott sorokat összehasonlítva rendezi a bemenetet.

sort –r	Fordított sorrendű rendezés. (reversre)
sort –n	Numerikus érték szerinti összehasonlítás. Alapértelmezett esetben alfabetikusan rendez. (numeric)

Tr parancs (karakterkonverzió)

A karakterkonverziós programok közül a leghasznosabb és legismertebb a **tr** (translate). Ez a parancs a standard bemenetről a standard kimenetre dolgozik, miközben az input egyes karaktereit kicseréli. A cseréhez két "táblázatot" használ, amelyeket string (karakterlánc) formában a parancs argumentumaiként kell megadni. Fontos megjegyezni, hogy nem karakterpárost töröl, kizárólag karaktereket, valamint hogy a standard ki- és bemenetet használja, fájlparamétert nem fogad el.

tr –d SET	Törli a bejövő karakterek közül azt, amelyik benne van a SET-ben, nincs cserélés. (delete)
tr –c SET	Kicseréli a SET-et az ő komplementerével (az összes karakter, ami nincs a SET-ben). (complement)
tr –s SET	Karaktersorozatot cserél ki egy karakterrel, miután elvégezte a cserélést vagy a törlést. Lecserél minden ismétlődő karakter-sorozatot, ami benne van a SET-ben, a karakter egyszeri előfordulásával. (squeeze-repeats)

Paste parancs

A **paste** parancs sorban kiírja minden megadott fájl sorait <TAB> karakterekkel elválasztva és újsor karakterrel lezárva. Ha nem adunk meg fájlnevet, akkor a standard bemenetet használja.

paste –s	A sorokat először egy fájlból veszi, ahelyett hogy minden fájlból venne egy-egy sort. (serial)
paste –d	Az elválasztó listában megadott karaktereket használja sorjában a <TAB> helyett az összefűzött sorok elválasztásához. Amikor az elválasztó listát kimerítette, elölről kezdi újra. (delimiters)

Cut parancs

A **cut** parancs a bemeneti fájl sorainak megadott részeit írja ki. Amennyiben a bemeneti fájlnév nem adott vagy az a standard bemenetet dolgozza fel. A részek megadása az opciókon keresztül történik.

cut –c	Csak a character-list által megadott karaktereket írja ki. (characters)
cut –f	Csak a field-list által megadott mezőket írja ki. A mezőket alapértelmezés szerint `TAB' karakterek választják el. (fields)
cut –d	Az -f opció által használt mezőelválasztót adja meg a delim első karaktere. (delimiter)
cut –s	Nem írja ki azokat a sorokat, melyek nem tartalmazzák a megadott mezőelválasztó karaktert. (only-delimited)

Spell parancs

A helyesírást ellenőrzi. A **spell** parancs a kimenetre csak a helytelenül írt szavakat írja ki. Természetesen megadható a nyelv is, amely alapján a helyesség vizsgálatát végzi. Alapértelmezetten angol nyelvű az ellenőrzés.

Rev parancs

A **rev** (reverse) parancs a bemenet sorainak tartalmát fordítja meg, soronként.

Tac parancs

A **tac** (cat visszafelé) parancs a annyiban különbözik a rev parancstól, hogy ez a teljes állományt megfordítja, azaz az állomány korábban első karaktere az utolsó karakter helyére kerül.

XI. Reguláris kifejezések

A ^ és $ karakterek

A ^ jel a sor elejére a $ jel a sor végére illeszti a mintát. Például a ^ $ kifejezés az üres sorra illeszkedik, a ^[^0-9]*$ kifejezés pedig a számot nem tartalmazó sorra.

Karakterosztályok

A **szögletes zárójel**be tett karaktersorozat illeszkedik az abban a pozícióban lévő bármely, a zárójelben felsorolt karakterre. A karakterek felsorolására érvényes szabályok az alábbiak:

- Kódjukat tekintve egymás után következő karaktereket rövidíteni lehet a **kötőjel** használatával. Például 0-9a-z jelenti az összes számjegyet és az angol ábécé összes kisbetűjét.
- A nyitó zárójelet követő ^ jel, a felsorolt karakterek tagadása. Azaz [^0-9] jelenti bármely, nem szám karaktert.

A **pont** olyan karakterosztály helyett áll, ami bármire illeszkedhet. Például 1991.03.10. lehet 1991.03.10 vagy 1991k03k10.

A **virgula** lehetővé teszi, hogy több kifejezést olyan kifejezésre szervezzünk, amelyik bármely önálló részkifejezésre illeszkedik.

Például:

- ”s[öe]rt” ~ ”sört|sert”
- ”s[ö|e]rt” **helytelen**
- ”sö|ert” **jelentése** vagy ”sö” vagy ”ert”

A \< és >\ karakterek

A \< karakterpáros a szó kezdetének pozícióját, míg a >\ karakterpáros a szó végének pozícióját jelöli.

A ?, * és + karakterek

A **kérdőjel** jelzi, hogy a megelőző kifejezés csak 0 vagy 1 esetben fordulhat elő. Például, a **colou?r** minta összeillik a color és a colour jelsorozatok közül bármelyikkel.

A **csillag** jelzi, hogy a megelőző kifejezés akárhány esetben fordulhat elő (beleértve a nullát is). Például, **go*gle** mintával összeillik a ggle, a gogle, a google stb.

A **plusz** karakter jelzi, hogy a megelőző kifejezés legalább 1 esetben fordulhat elő. Például a **go+gle** mintához illeszkedik a gogle, google stb. (de a ggle nem!).

Illeszkedések definiált korlátai

Néhány szabály a minta előfordulásának számosságára utal. Az egykarakteres reguláris kifejezést követő **{m,n}** sorozat, ahol m és n 256-nál kisebb, nem negatív egész, azt mondja, hogy a minta legalább m-szer és legfeljebb n-szer fordul elő egymás után. Ha csak az n szám van a zárójelpároson belül, annak jelentése, pontosan n-szer előforduló minta, ha m, van a zárójelpáron belül, annak jelentése legalább m előfordulás.

Zárójelek és visszautalások

A **\n** kifejezés (ahol **n** egy szám) a zárójelezéssel kijelölt mintára hivatkozik, mégpedig a kijelölés sorrendjében. Így a **^\(.\)\(.\).*\2\1$** minta olyan sorokra illeszkedik, ahol a sor első két karaktere tükörszimmetrikus az utolsó két karakterre (pl. 'axc23xa').

Maszkolás

Előfordul, hogy a mintában egy olyan karaktert keresünk, amely alapértelmezetten speciális reguláris karakternek értelmezendő. Ilyenkor használjuk a **** karaktert, amelyet a megfelelő karakter elé írva elérjük, hogy az adott karaktert ne tekintsük reguláris kifejezésnek. Pl.: **\.**

Grep, egrep, fgrep

A **grep** (global | regular expression) a megnevezett bemeneti fájlokban a megadott mintához illeszkedő sorokat keres. Amennyiben nincs bemenő fájlnév megadva, a standard bemenetet olvassa. Alapértelmezés szerint grep a mintához illeszkedő sorokat kinyomtatja.

grep –i	Ugyanazt teszi, mint az előző, csak nem tesz különbséget a kis és nagy betűk között. (ingore)
grep –E	A mintát bővített szabályos kifejezésként kezeli. (extended regular expression)
grep –F	A mintát fix stringekből álló listának veszi, melyek újsor-jelekkel vannak elválasztva, és amelyekből bármelyikhez való illeszkedést keressük. (fixed)

Az **egrep** hasonló a grep –E opciójához, de nem teljesen ugyanaz.

Az **fgrep** azonban teljesen azonos a grep –F opciójával.

XII. AWK

Az **awk** egy általános célú programozási nyelv, amelyet szöveges állományok földolgozására terveztek. Elnevezése a megalkotói – **A**lfred Aho, Peter **W**einberger és **B**rian Kernighan – családnevének kezdőiből született. A UNIX 3 verziójában jelent meg. Ideális szöveges állományok szűrésére, átformálására, kiértékelésére. Működésének elve, hogy szövegsorokat választ ki az állományból és azon műveleteket hajt végre. Ma is minden unix rendszeren van legalább egy awk változat. Az awk kizárólag karaktersorozatokkal, más szóval sztringekkel foglalkozik, a számábrázolásra is ezeket használja. Reguláris kifejezések és minta alkalmazása is lehetséges. Ezen felül parancs módban is használható.

XIII. SED (Stream Editor)

A **sed** egy nem interaktív szövegszerkesztő program. Bemenetét a standard inputról várja, kimenetét pedig a standard outputra írja. Működésének elve, hogy beolvassa a bementről a sorait, és mindegyiken végrehajtja a megadott szerkesztési parancsokat. A műveleteket a bemenet elején kezdi el végrehajtani.

Sed opciók

sed –n	Elnyomja az alapértelmezett kimenetet.

Sed szerkesztő parancsok

▪ a	Sor beszúrás a tartomány mögé		
▪ i	Sor beszúrás a tartomány elé		
▪ p	Kiíratás	sed –n ’4p’ fajl	Kiírja a fajl 4. sorát
▪ s	Csere		
▪ d	Törlés	sed –n ’/^$/d’ fajl	Törli fajl-ból az üres sorokat
▪ n	Következő sor	sed –n ’2~2p’ fajl	Kiírja a fajl páros sorait
▪ q	Kilépés		
▪ !	Negáció		

XIV. Folyamatok időzítése

At

Az **at** parancs segítségével tudunk elindítani egy utasítást a megadott időpontban. Az időpont megadásakor elég laza szabályok vannak, az at parancs elég intelligens ahhoz, hogy a legtöbb angol időpont- és dátummegadási módot felismerje. Lehetőség van a tomorrow, today, midnight, now stb szavak használatára is.

at –l	Kilistázza az aktuális at jobokat. (~ **atq**)	**Kulcsszavak az időzítéshez**
at –r jobszam	Törli a megadott at jobot. (~ **atrm**)	am, pm, now, noon, midnight, today, tomorrow

Például:

▪ At 03:21 am Jun 21 script	Június 21-én fog lefutni hajnali 3 óra 21 perckor a script nevű program.

Crontab

A **crontab** hasonló funkcióval bír, mint társa az at, azonban a crontab a folyamatok periodikus ismétlését teszi lehetővé. Használatakor 6 mezőt kell megadni:

- 1. mező 0-59 (percek)
- 2. mező 0-23 (órák)
- 3. mező 1-31 (napok)
- 4. mező 1-12 (hónapok)
- 5. mező 0-6 (napok, 0 = vasárnap)
- 6. mező Parancs

A * karakter az adott intervallum minden egyes értékét felveheti.

crontab –e	Az adott crontab tartalmának szerkesztése. (edit)
crontab –l	Kilistázza az aktuális crontab jobokat. (list)
crontab –r	Törli a megadott crontab jobot. (remove)

Például:

- 0 0 * * 0 script Minden vasárnap 00:00-kor lefut a script program.
- 0 2 5 * * script Minden hónap 5-én 00:02-kor lefut a script program.

Sleep

A **sleep** parancs segítségével megadott hosszúságú késleltetés indítható. Alapértelmezetten másodpercben méri a késleltetési hosszúságot.

Például:

- sleep 5; script 5 másodperc múlva lefut a script program.
- sleep 2h; script 2 óra múlva lefut a script program.

Time

A **time** parancs segítségével a folyamatok időfelhasználásáról informálódhatunk.

Például:

- time sort < nagyfajl Kiírja a standard outputra, hogy a nagyfajl-t mennyi idő alatt rendezte abc sorrendbe.

XV. Shell scriptek

Egy shell script nem más mint shell-parancsok sorozata, amelyeket az újrafelhasználás jegyében egy file-ba írunk. Ahhoz, hogy egy shell scriptet létrehozzunk, az **sh** parancsot kell használjuk, majd a létrehozott fájlnak futtatható jogot kell adnunk (775).

Változók

Egy változónak a következőképpen adhatunk érteket:	valtozo = ertek	**now = `date`**
A változó értekét mindegyik shellben az alábbi módon kapjuk vissza:	echo $valtozo	Speciális idézőjel (altgr+7)

Argumentumok

Speciális változók, amelyek a script meghívásának pillanatában kapnak értéket. Az argumentumokat **$** jellel jelöljük és 1-től számozzuk. A **$0**-s értékű argumentum a parancsot jelöli, míg a **$*** argumentum az összes argumentumot. Példa argumentumadásra: **./script alma korte**

XVI. Test parancs

Ha a feltétel igaz volt, akkor 0-val tér vissza. Használhatjuk numerikus értékek tesztelésére (eq = eqvivalent, gl = greater than, lt = lower then), fájlt típusok kezelésére (s = scan, f = file, d = directory, w = write, r = read), például: **test –d mappa**. Használható még karakterláncok tesztelésére (=,!=), vagy logikai összehasonlításokra (a = and, o = or), például: **test –w $file –a –r $file**.

XVII. Vezérlési szerkezetek

Elágaztatás

```
if  feltétel
then
        parancsok
elseif
        parancsok
else
        parancsok
fi
```

Az if az őt követő parancsok közül az utolsónak az exit statusát nézi, ha az nulla (vagyis a parancs sikeresen futott), akkor "igaz"-nak veszi, különben "hamis"-nak. (C programozási nyelvben ez pont fordítva van: a nulla felel meg az "hamis"-nak, minden más meg az "igaz"-nak).

For ciklus

A for ciklus némiképp különbözik a Pascal es C nyelvek for ciklusától, mégpedig abban, hogy csak egy rögzített lista elemein lehet végigmenni vele. Tipikus felhasználása, hogy bizonyos kiterjesztésű file-okon vagy a parancssor argumentumain megyünk végig.

While ciklus

A while ciklus addig fut, amíg a hasában levő feltétel igaz, az until pedig pont fordítva: amíg a feltétele igaz nem lesz. Mindkét esetben do és done közé kell zárni a ciklus parancsait.

XVIII. Aritmetikai műveletek

Az **expr** parancs segítségével értelmezhetünk különböző típusú aritmetikai műveleteket. Az expr kifejezésként kiértékeli az argumentumait, majd elvégzi a rajtuk értelmezett műveletet (+ , - , * , /, %). Fontos megjegyezni, hogy * művelet esetében jelölni kell, hogy nem a minden karaktert jelölő meta karakterről van szó, ezért a szorzás így néz ki: **expr 2 * 3** vagy **expr 2 "*" 3**.

XIX. Hálózati alapok

Whoami

A **whoami** parancs a pillanatnyilag hatályban levő felhasználói azonosító (user ID) alapján kinyomtatja a felhasználói nevet.

Who

Amennyiben az opciókon kívül nincs argumentuma, a **who** parancs kinyomtatja minden, pillanatnyilag bejelentkezett felhasználóról a következő információkat:

- Bejelentkezési név (login name)
- Terminál vonal (terminal line)
- A bejelentkezés ideje (login time)
- Távoli gépnév vagy X kijelző (remote hostname or X display)

W

A **w** parancs információkat jelenít meg arról, hogy éppen hány felhasználó van a gépen és hogy mit csinálnak. A fejléc megmutatja - ebben a sorrendben - az időt, mióta működik a rendszer, jelenleg hány felhasználó van belépve és a rendszer átlagos terhelését az elmúlt 1, 5 és 15 percben.

Users

A **users** parancs egy sima szóközzel elválasztva jeleníti meg az aktuális hosztra bejelentkezett felhasználói nevek listáját. Minden kiírt felhasználói név megfelel egy-egy bejelentkezési viszonynak, így ha egy felhasználó egynél többször van belépve, akkor annyiszor íródik ki a login neve, ahányszor bejelentkezett.

Finger

A **finger** a rendszer felhasználóiról mutat információkat.

finger **–s** A finger megmutatja a felhasználó belépési nevét, valódi nevét, terminálját és hogy az írható-e (a terminál neve mögött ``*'' jelenik meg, ha nem írható), mióta nem csinált semmit, mikor lépett be, valamint irodájának helyét és telefonszámát. A belépés idejét hónap, nap, óra, perc formában adja meg, kivéve ha hat hónapnál régebben lépett be; ez esetben az óra és a perc helyett az évet jelzi ki. Az ismeretlen eszközök és a nem létező belépési valamint nyugalmi időt csillaggal jelzi.

finger **–l** Többsoros megjelenítés, amely magában foglalja az -s kapcsoló által mutatott adatokat, valamint a felhasználó home mappáját, otthoni telefonszámát, belépési shelljét, leveleinek állapotát és a home mappájában található .plan , .project valamint .forward nevű fájlok tartalmát.

Write

A **write** parancs lehetővé teszi a többi felhasználóval való kommunikációt úgy, hogy az általunk beírt sorokat megjeleníti az ő terminájukon. Amikor elindítjuk a write parancsot, a felhasználó, akinek írunk, a következő üzenetet kapja:

```
Message from yourname@yourhost on yourtty at hh:mm ...
```

Amit ezek után beírunk, az a megadott felhasználó terminálján fog megjelenni. Ha válaszolni akar, neki is el kell indítania a write parancsot. Ha készen vagyunk, le kell ütni a sor vége vagy a megszakító karaktert. A másik felhasználó egy 'EOF' üzenetet fog látni, ami jelzi számára, hogy a társalgásnak vége.

Talk

A **talk** egy olyan vizuális kommunikációs program, amely terminálunkról sorokat másol egy másik felhasználóéra. Ha mi hívunk először, a talk program a következő üzenetet küldi:

```
Message from TalkDaemon@ő_gépe...
talk: connection requested by mi_nevünk@mi_gépünk.
talk: respond with: talk mi_nevünk@mi_gépünk
```

Ekkor a címzettnek a következő parancs begépelésével kell válaszolnia:

```
talk mi_nevünk@mi_gépünk
```

Amint a kommunikáció engedélyezve lett, a benne résztvevő két felhasználó egyszerre gépelheti üzenetét, amely üzenetek két egymástól elválasztott ablakban jelennek meg a képernyőn. Kilépéshez egyszerűen be kell írni az interrupt-karakterünket, a talk ezután a kurzort a képernyő aljára mozgatja és visszaállítja a terminál eredeti helyzetét.

Mesg

A **mesg** parancs segítségével letilthatjuk, illetve engedélyezhetjük a write és talk üzenetek fogadását. Engedélyezés: **mesg –y**. (y = yes, n = no)

Telnet

A **telnet** parancs egy másik számítógéppel történő interktív kommunikációra használatos. Magyarul ~ távoli bejelentkezés.

Ssh

Az **ssh**-t (secure hell) egy helyi és egy távoli számítógép közötti biztonságos csatorna kiépítésére fejlesztették ki. Működésének elve, hogy nyilvános kulcsú titkosítást használ a távoli számítógép hitelesítésére. Használata: **ssh felhasznalo@host**

Ftp

Az **ftp** parancs segítségével állományátvitelt biztosíthatunk egy távoli számítógéppel. A kapcsolat létrejötte után az alábbi, fontosabb parancsokat használhatjuk:

- user Azonosítja a felhasználót a távoli FTP szerveren.
- cd Lépegetés a távoli számítógép mappái között.
- lcd Lépegetés a helyi számítógép mappái között.
- pwd A távoli számítógép aktuális könyvtárának kiírása.
- lpwd A helyi számítógép aktuális könyvtárának kiírása.
- get Letöltés.
- put Feltöltés.
- quit Az ftp kapcsolat lezárása.

Az **sftp** az ftp titkosított verziója.

XX. VI (Visual Editor)

„Ahol UNIX van, ott VI is van.”

A VI működése

A szerkesztendő állományt egy bufferbe teszi, az ezen végzett változtatásokat pedig folyamatosan naplózza.

A VI 3 üzemmódja

- Parancs mód
- Beviteli mód
- Ex mód

Parancs mód

A leütött billentyűket parancsként értelmezi és hajtja végre.

Beviteli mód

Beviteli módba akkor kerül a VI, ha az a A i I o O c C s S R parancsok valamelyikét adtuk ki. Beviteli módban a begépelt szöveg a szerkesztett állományba kerül, s ilyenkor semmilyen parancsot nem lehet kiadni mindaddig, amíg a beviteli módból vissza nem lépünk parancsmódba, rendesen az ESC billentyű lenyomásával.

Ex mód

Ex módba akkor kerül a VI, amikor a : / ? ! karakterek valamelyikét ütjük le parancsmódban. Ekkor az ex editornak szóló parancsokat gépelhetünk be, a sorlezáró újsor karakter hatására a begépelt parancsokat az ex végrehajtja. A parancs végrehajtását a DEL billentyűvel szakíthatjuk meg. Az ex módból parancs módba kerülünk vissza.

Pozicionálás karakterek alapján

h, <BACKSPACE>	A kurzort egy karakterrel balra mozgatja.
l, <SPACE>.	A kurzort egy karakterrel jobbra mozgatja
$	A kurzort a sor utolsó karakterére állítja.
0,	A kurzort a sor első karakterére állítja.
n	A kurzort a sor n-edik oszlopára állítja.
^	A kurzort a sor első nem blank karakterére állítja.
fx	A kurzort jobbra mozgatja az első x karakterre.
Fx	A kurzort balra mozgatja az első x karakterre.
tx	A kurzort jobbra mozgatja, az első x karakter elé.
Tx	A kurzort balra mozgatja, az első x karakter mögé.
;	A megelőző f, F, t, T parancsok által keresett karakter következő előfordulását keresi.
,	Az f, F, t, T parancsok által előzőleg keresett karakter újabb előfordulását keresi visszafelé.

Az összes itt felsorolt parancs a kurrens sorban működik, beleértve a **t** és **f** parancsokat is.

Pozicionálás sorok alapján

J	A kurzort egy sorral lejjebb mozgatja.
K	A kurzort egy sorral feljebb mozgatja.
+, <CR>	A kurzort lefele mozgatja, a következő sor elejére.
-	A kurzort felfele mozgatja, a felette lévő sor elejére.

Pozicionálás szavak alapján

w, W	A kurzort a következő szó első karakterére mozgatja.
b, B	A kurzort a megelőző szó első karakterére viszi.

Pozicionálás mondatok alapján

(	A kurzort a mondat elejére viszi.
)	A kurzort a következő mondat elejére viszi.

Pozicionálás bekezdések alapján

{	A kurzort a bekezdés elejére viszi.
}	A kurzort a következő bekezdés elejére viszi.

Pozicionálás az állományban - görgetés

^f	Teljes képernyőnyit előre görget.
^d	Fél képernyőnyit előre görget.
^b	Teljes képernyőnyit hátrafele görget.
^u	Fél képernyőnyit hátrafele görget.

Pozicionálás mintaillesztéshez

/minta	Keresés előre.
?minta	Keresés visszafelé.
n	Az utolsó minta keresés ismétlése.
N	Az utolsó minta keresés ismétlése, de azzal ellenkező irányba.

Törlés parancsmódban

x	Törli az adott pozícióban a karaktert.
dx	A kurzortól kezdve törli az x szövegobjektumot. Például db a kurzortól visszafele a legközelebbi szó elejéig töröl, d$ a sor végéig, d(a kurzortól az aktuális mondat elejéig, d} az aktuális bekezdés végéig, 3d{ a kurzortól visszafele három bekezdést.
dd	Törli az aktuális sort.

Pozícionálás beviteli módban

a	Beviteli módba vált és a szöveget a kurzor mögé állítja.
A	Beviteli módba vált és a szöveget a sor mögé állítja.
i	Beviteli módba vált és a szöveget a kurzor elé állítja.
I	Beviteli módba vált és a szöveget a sor elé állítja.
o	Beviteli módba vált és a új sort tesz az adott sor után.
O	Beviteli módba vált és a új sort tesz az aktuális sor után.

Kilépés parancs módban

ZZ	Mentés és kilépés
u	Visszavonás (undo).

Mentés és kilépés Ex módban

:wq	Mentés és kilépés
:w file	File néven elmenti az állományt.
:w! file	Írásvédelem esetén is ment.
:q!	Kilépés.