

Dr. Mileff Péter

Unix/linux operációs rendszer üzemeltetése

1. Előadás

Bemutkozás és követelmények

- Dr. Mileff Péter - Általános Informatikai Tanszék
 - 111. szoba.
 - Konzultációs idő: szerda12-14.
- Követelmények:
 - Vezetett gyakorlat nincs.
 - Jelenléti ív nincs.
 - 1 zárthelyi dolgozat a félév vége felé.
 - 10. hét
 - Zárás:
 - Aláírás + kollokvium (írásbeli és szóbeli)

2

Ajánlott irodalom

- Saját segédlet:
 - <http://www.iit.uni-miskolc.hu/~mileff/linux>
 - A teljes tananyagot tartalmazza.
- KÖNYV:
 - PERE LÁSZLÓ, **GNU/LINUX rendszerek üzemeltetése I-II**. Kiskapu Kiadó. 2005.
 - Gagné, Marcel, **Linux rendszerfelügyelet**. Kiskapu Kiadó. 2002.
 - Sikos László, **Bevezetés a Linux használatába**. BBS-INFO Kiadó. 2005.
 - stb

3

A Unix/Linux története...

4

Bevezetés

- A Unix nem egy új operációs rendszer
 - Régóta egyre növekvő arányban jelen van a számítástechnikai világban
 - Hosszú ideig az egyetemi, kutatói szférában volt egyeduralgoló
- Egyre újabb és újabb területeket hódít meg
 - banki, vállalati, adatfeldolgozó szféra
- Miben rejlik a legfőbb ereje?
 - Dinamikusságában, alkalmazkodóképességében.
 - Képes ugyanazt a környezetet nyújtani:
 - Mind a mainframe nagyépeknek,
 - mind az otthoni PC-k számára.
- Manapság egyre inkább szükség van
 - egy olyan környezetre, amely képes hardvertől, platformtól függetlenül mindenhol ugyanazt nyújtani.

5

A Unix rövid története

- Első változatát 1969-ben készítette Ken Thomson és Dennis Ritchie
 - az AT&T Bell Laboratóriumában egy PDP-7 típusú számítógépre.
- 1973: a rendszermagot átírták C nyelvre
 - Ennek köszönheti a Unix/Linux a legnagyobb előnyét ma is, a hordozhatóságot.
- Az AT&T kezdetben ingyen az amerikai egyetemek rendelkezésére bocsátotta a Unix forráskódját
 - Tíz éven belül százezer fölé emelkedett a működő Unix rendszerek száma.
 - A gyors terjedés miatt:
 - nem volt egységes ellenőrzése senkinek sem a forráskód, a rendszer egysége felett.

6

A Unix rövid története

- Számos (helyi) változat alakult ki, amelyek közül a két legjelentősebb:
 - Berkeley egyetemen kifejlesztett BSD Unix,
 - AT&T „hivatalos” változata a System V (System Five - SVR4)
 - Ma is számos alváltozat van forgalomban.
- A Unix egyre népszerűbbé kezdett válni a kereskedelmi szférában
 - egyre több cég ismerte fel egy egységes Unix szabvány fontosságát
 - több egységesítő, szabványosító bizottság és csoportosulás kezdett dolgozni
- IEEE kidolgozta a „**POSIX**” (Portable Operating System Interface (x)) ajánlást, amely igyekszik egyesíteni a két fő irányt.

7

Elterjedt UNIX-ok

A UNIX neve	Gyártó cég
AIX	IBM
HP-UX	Hewlett-Packard
Irix	Silicon Graphics
Nextstep	Next
Solaris	Sun Microsystems
SunOS	Sun Microsystems
Unixware	Novell

8

A GNU projekt, avagy hogy kerül ide a Linux...



9

A Linux előzménye

- Amikor a Unix még csak az egyetemi és akadémiai szférában volt közismert:
 - kialakult körülötte egy hatalmas programkörnyezet
 - minden egyetem, kutatóintézet elkészítette saját megoldásait problémáira
 - szövegszerkesztés, mindenféle apró utility, fordítóprogramok.
 - Az intézmények non-profit szervezetek voltak, elkészült szoftvereiket publikussá tették:
 - nem volt egységes ellenőrzése senkinek sem a forráskód, a rendszer egysége felett.
- Az egységes C nyelv és a környezet miatt:
 - minden Unix felhasználó lefordíthatta, használhatta, módosíthatta és továbbfejleszthette őket szinte teljes szabadsággal.
 - Ennek eredményeképpen alakult ki az **FSF**
 - **Free Software Foundation** - Richard Stallman

10

A Linux előzménye

- **FSF célja:** egy szabadon (forráskódban is) ingyen hozzáférhető szoftverkörnyezet biztosítása.
- Elindult a **GNU project** (GNU is Not UNIX):
 - egy minél teljesebb Unix rendszert kíván létrehozni és biztosítani.
 - Ennek jogi megfogalmazása a **GPL (GNU General Public License)**
 - GPL alá eső szoftvert bárki:
 - Készíthet, használhat, módosíthat, továbbadhat.
 - GPL szoftverért és módosításért pénzt kérni nem szabad
 - GPL forrás módosítva is GPL forrás marad.
 - Fel kell tüntetni a módosítás dátumát, módosító nevét, elérhetőségét, stb.

<http://www.gnu.org/>

11

A Linux

- Megvolt tehát a GNU környezet:
 - fordítók, segédprogramok, és a szabadon terjeszthető XFree grafikus felület.
- Egyedül egy operációs rendszer mag hiányzott:
 - amely bizonyítottan szabad.
 - Ennek megírását kezdte el helsinki egyetemista korában **Linus Torvalds**.
 - több száz segítőjével együtt létrehozta azt, amit ma Linuxként ismerünk:
 - egy teljes, szabad operációs rendszert bárki számára.
 - Így született meg a **GNU/Linux**.
 - Jogi értelemben nem UNIX.
 - POSIX szabványt követi főleg.



12

Mik a disztribúciók?

- Egy Linux kernelen alapuló teljes (működőképes) Unix rendszer
- Hogyan készül egy disztribúció?
 - a C forrásban meglévő utility-k, programok lefordításából, jegyzékstruktúrába helyezéséből és összekonfigurálásából áll.
- Sokféle disztribúció létezik, ingyenesek is és kereskedelmiek is. Néhány:
 - Slackware Linux, Mandriva Linux, OpenSuse, Redhat Linux, Novell Linux, Ubuntu, Kubuntu, Zenwalk, Frugalware Linux, Uhu Linux, Fedora Linux, stb.

13

Top 100 Linux

Hely	Disztribúció	H.P.D*	26	Chakra	294
1	Ubuntu	2235	27	Peppermint	291
2	Fedora	1474	20	KNOPIX	282
3	Mint	1373	29	ClearOS	282
4	openSUSE	1247	30	sidux	279
5	PCLinuxOS	1068	31	PC/OS	267
6	Debian	999	32	Ubuntu Studio	262
7	Mandriva	896	33	Salix	261
8	Sabayon	754	34	Xubuntu	247
9	Arch	734	35	Zentyal	242
10	Lubuntu	715	36	BackTrack	234
11	Puppy	697	37	gOS	227
12	Slackware	569	38	TinyMe	220
13	MEPIS	545	39	Frugalware	218
14	CentOS	519	40	Parrot Magic	217
15	FreeBSD	506	41	Super OS	216
16	Ultimate	478	42	EasyPeasy	213
17	Tiny Core	406	43	Slitaz	210
18	CrunchBang	403	44	Unity	208
19	Gentoo	384	45	Element	207
20	Zenwalk	375	46	Absolute	203
21	Dreamlinux	368	47	Clonezilla	197
22	Kubuntu	347	48	SystemRescue	194
23	PC-BSD	332	49	Zorin	191
24	Vector	322	50	Pardus	173
25	Red Hat	316	51	Macpup	173

www.distrowatch.com

14

A Linux működésének áttekintése...

- **A Linux egy valódi többfeladatos (multitask) és többfelhasználós (multiuser) rendszer.**
 - Mi volt a célja a 386-ossal?
 - A cpu nyújtotta fejlett tár és taszkkezelési lehetőségeket, valódi időosztásos környezetet kihasználását célozta meg.
- **Két üzemmód:**
 - **Védett mód:** kernel használja a mag futtatására. A kernelnek hozzáférése van a gép összes fizikai erőforrásához
 - **User mód:** a felhasználói folyamatok így futnak.
- Lehetőség van több, egymástól független „user” módú taszk definiálására.
 - egymástól védettek, nem tudják egymás és a felügyelő kernel memória- területét kiolvasni vagy módosítani,
 - a gép közvetlen hardver erőforrásaihoz sincs hozzáférésük

15

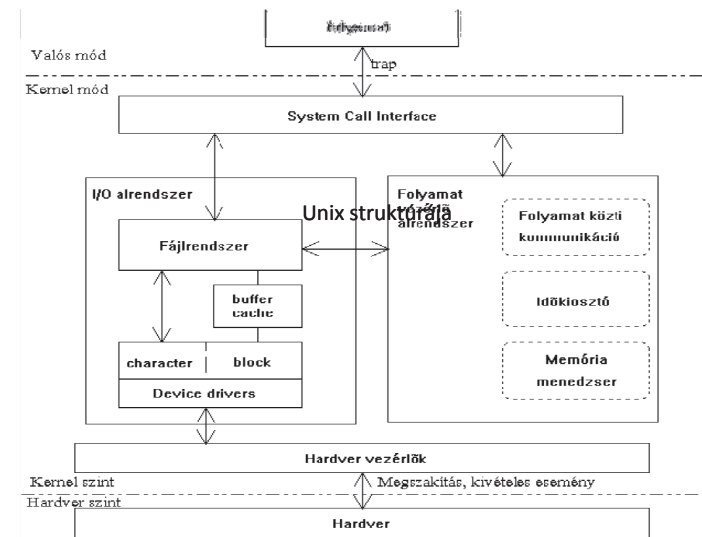
16

A 386-os csoda

- Így biztosítható az egyes felhasználói programok egymástól való védelme.
 - az egyes folyamatoknak a gép fizikai erőforrásaihoz (pl. winchester, képernyő) nincs közvetlen hozzáférésük,
 - bármilyen perifériaműveletet csak a kernel meghívása útján végezhetnek
- A kernel:
 - teljes mértékben, fizikai szinten hozzáfér a gép erőforrásaihoz.
 - Fizikai, a lehető legalacsonyabb szinten kezeli is a hardvert, a legnagyobb teljesítmény elérése érdekében

17

Unix/Linux struktúrája



18

Memóriakezelés

- Kihasználja a 386 által nyújtott lehetőségeket
 - lapozásos virtuális memóriakezelés
 - a fizikai memória kiegészítése a hdd-ről vett virtuális memóriával (page vagy swap terület)
- A teljes memóriát lapokra osztja
- Ezen virtuális lapokat rendeli hozzá az egyes folyamatokhoz
 - gondoskodik róla, hogy az éppen szükséges lapok a fizikai memóriában legyenek
- A Linux használja a virtuális tárkezelés mindkét fajtáját:
 - a **lapozást** (paging) és a **tárcserét** (swapping)

19

Memóriakezelés

- **Lapozás:**
 - a rendszer arra ügyel, hogy a szükséges lapok a fizikai memóriában legyenek
 - ha azok esetleg diszken vannak, akkor gondoskodik memóriába olvasásukról,
 - illetve ha a fizikai memória megtelt, akkor a ritkábban használt lapokat a diszke írja.
- **Tárcsere:**
 - a rendszer figyelemmel kíséri az egyes folyamatok aktivitását is,
 - ha szabad memóriára van szükség, egy inaktív folyamat egészét háttértárra írja,
 - Így felszabadítja a folyamat által használt összes fizikai memóriát

20

Memóriakezelés

- A Linux a két módszer keverékét használja:
 - amíg rendelkezésre áll elegendő memória, úgy csak egyes lapokat lapoz ki/be
 - Ha egy folyamat hosszú ideje inaktív, és nem csak egy-két lapnyi memóriára van szükség,
 - akkor az adott folyamathoz tartozó összes fizikai lapot diszkre menti.
- A hdd virtuális memóriakezelése dinamikus
 - menet közben is változtatható,
 - az operációs rendszer leállítása nélkül lehetőségünk van a virtuális memória méretének megváltoztatására.
 - A **swap** terület használható fájlként, vagy akár külön partíciónként. Akár egyszerre több swap terület is.

21

Buffer Cache

- A buffer cache a Unix rendszerek diszk-eléréshez használt gyorsítótárja
 - a kernel kezeli, mert minden folyamat csak a kernel meghívásával végezhet diszkműveletet
- **Célja:** az I/O hozzáférések gyorsítása, ezzel pedig a „felhasználói élmény” növelése.
- Mérete dinamikusan változik
 - a rendszer-terheléstől függően
- Minden diszk írás ezen keresztül történik
 - minden írás először a cache memóriába kerül,
 - vagy egy megadott idő elteltével íródik ki diszkre, vagy pedig akkor, ha a rendszer számára „elegendő” kiírnivaló összegyűlt

22

Miért fontos a leállítás?

- Kikapcsolás előtt mindig szükséges a diszk tartalmának szinkronizálása a memóriában lévő állapottal
 - ezen lépések elmulasztása esetén kikapcsoláskor a diszk tartalma helytelen lehet.
- Ezzel fizetünk a nagyobb teljesítményért.
- Az adatvesztés veszélye minden, a diszk-írást bufferelő rendszerben fennáll.
 - pl.: Unix, Linux, Windows, Haiku, stb.

23

További gyorsítási megoldások...

24

Demand Paging

- **Demand paging:** („igény szerinti lapozás”)
 - egy futtatható fájl végrehajtásakor nem az egész fájl töltődik be a memóriába,
 - mindig csak azok a lapjai, amikre a végrehajtás során éppen szükség van.
- A sebességnövekedés:
 - Minden programnak vannak olyan részei melyek csak egyszer (vagy akár egyszer sem) futnak le
 - ezeket a részeket vagy be sem tölti a rendszer, vagy miután lefutottak felszabadítja az általuk elfoglalt memóriaterületet.

25

Osztott kódkönyvtárak használatának alapelve

- **Alapelv:**
 - a programok nagy része C nyelven íródnak,
 - valószínűleg sokban van olyan függvény, amely más programokban is előfordul.
 - Ezeket felesleges lenne minden programmal a memóriába tölteni, elég egyszer.
 - Meg kell mondani a programoknak, hogy hol keressék ezeket a függvényeket a memóriában.
 - Ezt csinálja a dinamikus linker:
 - A programokba beépített programrészletnek segítve gondoskodik a függvények megtalálásáról
 - illetve a memóriába töltésükről, amennyiben még nem lennének betöltve

26

Copy-on-write mechanizmus

- **Copy-on-write mechanizmus:**
 - új folyamat létrehozása mindig egy másik folyamat „memóriájának lemásolásával” történik.
- **Gyorsítás:**
 - Mivel viszont egy memórialapra több folyamat memóriatérképéből lehet hivatkozni, nem kell azt a lapot lemásolni,
 - csak el kell helyezni a lapra mutató hivatkozásokat a megfelelő helyeken.
 - Csak arra kell vigyázni, hogy amikor az ugyanarra a lapra hivatkozó folyamatok közül valamelyik módosítani akarja a lapot,
 - akkor le kell másolni a számára, és így már módosíthatja, mert az már csak az övé

27

A folyamatok ütemezése...

28

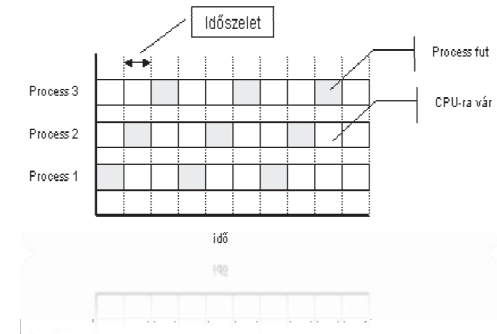
Folyamatok ütemezése

- Az operációs rendszernek egy CPU-n belül kell konkurensen több feladatot végrehajtania
 - ezért valamilyen formában meg kell osztania a rendelkezésre álló CPU időt az egyes folyamatok között.
- A Unix alapú rendszerek a **preemptív időosztásos ütemezés** módszerét alkalmazzák:
 - a rendelkezésre álló időt felosztja egyenlő részekre,
 - és ezekből az egyenlő időszelletekből juttat – a folyamat prioritásának megfelelően – többet vagy kevesebbet az adott folyamatnak.
 - Ha az adott folyamat számára kijelölt időszellet letelt, a kernel megszakítja a folyamat futását
 - és más folyamatnak adja át a vezérlést

29

Folyamatok ütemezése

- Linuxban az ütemezés alapegysége az 1/100 másodperc.



30

Folyamatok ütemezése

- **A Unix(Linux) nem valós-idejű (real-time) operációs rendszer:**
 - ha több folyamat fut egyszerre, és az egyiktől elkerül a vezérlés, akkor valamekkora idő múlva vissza is fogja majd kapni
 - a két aktív (futó) állapot közti időre azonban nincs szigorú felső korlát.
 - Az esetek 99.9999999 százalékában ez az idő (még egy leterhelt rendszeren is) pár tized másodperc –
 - azonban soha nem mondhatjuk, hogy biztosan csak ennyi.

31

Köszönöm a figyelmet!

32