Számítógép architektúrák

Felhasználói felületek

Általános
INFORMATIKAI
Tanszék

Felhasználói felületek, kezelők

- User Interface (UI), Command Languege (CL) stb. elnevezések is
- Céljuk:
  - Ezekkel „kezeljük” a rendszert,
  - Manipulálunk eszközökön, fájlokon
  - Indítunk programokat (processzeket készítünk)
  - Input adatokat adunk meg, eredményeket jelenítünk meg
- Alapvető nyelveik:
  - Parancsnyelv – válasznyelv

(Az egyes alkalmazásoknak is vannak felületeik ...)

Szokásos két osztályuk

- (Alfanumerikus) parancsnyelv értelmezős
  - Régebbi
  - Hatékonyabb
  - Kisebb erőforrás igényű
- Grafikus felületek (Graphical User Interface, GUI)
  - Ez az újabb
  - Kényelmes, felhasználóbarát
  - Nagyobb erőforrás igény

Parancsértelmezős felületek

- **A modell:**
  - **Adott egy (alfanumerikus) terminál (konzol és drivere)**
  - **Adott egy parancsértelmező processz**
    - **Készenléti jelet (prompt) ad a konzolra,**
    - **Parancsot (csövet, parancslistát) beolvas, értelmez, átalakít és végrehajt, vagy végrehajtat.**
  - **A parancsok valójában kérelmek, melyeket az OS magjának (kernel) szolgáltató rutinjai, vagy önálló processzek szolgálnak ki**

Egy parancsnyelv, a Bourne shell

- A Unix OS parancsnyelve
  - A Unix (szerű) OS-ek nagyon elterjedtek (sok könyv, ismertető)
  - Parancsnyelveik (különösen a Bourne shell) egyszerűek
- Miért a Bourne burok? Stephen Bourne 1977

Neve	Programja	Szokásos promptja
Bourne shell	/bin/sh	$
Korn shell	/bin/ksh	$
C shell	/bin/csh	%
Bourne again shell	/bin/bash	$ Brian Fox1987

A burok processz

- Önálló entitás, azonosítója a pid (process identification number)
- A /bin/sh (vagy /bin/bash) program fut benne
- Van 3 nyitott adatfolyama
  - A 0 leírójú stdin (szabványos bemenet), ahonnan a parancsokat, csöveket, parancslistákat olvassa.
  - Az 1 leírójú stdout (szabványos kimenet), ahová az eredményeit írja.
  - A 2 leírójú stderr (szabványos hibakimenet), ahová a hibaüzeneteit írja.
- A nyitott adatfolyamok „szokásos módon” eszközökhöz vannak kapcsolva

A burok processz működése

- Az stdout csatornájára kiírja a készenlét jelet (prompt), jelezve, hogy parancsot, csövet, parancslistát vár
- Az stdin csatornáján parancsot, csövet, parancslistát olvas be,
  - Azt elemzi, értelmezi,
  - átalakítja, majd végrehajtja, vagy végrehajtatja.
- A végrehajtás eredményét az stdout, ill. stderr csatornára írja, végül visszatérési értéket produkál.

A visszatérési érték

- Lehet normális (0),
- lehet nem normális (nem 0), ennek oka többféle
  - valami hiba van,
  - nincs hiba, de szemantikailag van gond.

  (Pl. grep szűrő nem talál minta-egyezést, vagy „test” parancs tesztelése nem igaz (logikai igaz/hamis).)
- A visszatérési értéket a programvezérlésben használhatjuk majd.

A parancs fogalma

- **Fehér karakterekkel határolt szavak sora**
  - **első szó a parancs neve,**
  - **többi szó az argumentumok.**
- **Az sh beolvassa, értelmezi, átalakítja, végrehajtja**
  - **saját maga (belső p.),**
  - **gyermek processzben (külső p.)**

Mindkét esetben van visszatérési értéke!

Vannak szabványos adatfolyamok!

Egy példa parancsra

> **find . -name a.c -print**

ahol a szavak számozása

0 1 2 3 4

Azaz a fenti parancs 5 szóból áll. Vegyük észre, hogy a burok promptja nem része a parancsnak! Inputot nem kíván, outputja (esetleges hibaüzenetei is) a képernyőre megy. Vajon mi a visszatérési értéke? És ez?

> **find . -name a.c -print >myfile.txt**

Parancsokat kell tanulni ...

- **Legfontosabb dokumentum az on-line kézikönyv, a man (On-line Manual)**

  > man [-opc] [section] lap
- **Tudni kell angolul ...**
- **Sajnos, nincsenek „dzsókerek”, a fontos parancsok nevét pontosan kell tudni!**
- **Érdemes „parancs kártyát” készíteni, a fontos parancsok nevével, rövid leírásukkal.**

Parancsok: editorok

- ed sororientált
- vi (vim) képernyő orientált
- mcedit képernyő orientált
- pico, nano egyszerű, sok helyen (vt100 kell)
- joe
- stb.

Parancsok: kiírók

- **cat** összefűz, stdout-ra
- **pr** nyomtat, stdout-ra
- **head** fájl első sorait, stdout-ra
- **tail** fájl utolsó sorait
- **more, less** lapokra tördelő szűrő
- **od** oktális ömlesztés (dump)

Parancsok: jegyzékekkel kapcsolatban

- **ls** jegyzék tartalom lista (dir helyett)
- **mkdir** jegyzék készítés
- **rmdir** jegyzék törlés
- **cd** munkajegyzék váltás
- **pwd** munkajegyzék lekérdezés
- **chmod** fájl védelmi maszk váltás (Nemcsak jegyzékre)
- **chown** fájl tulajdonos váltás (Nemcsak jegyzékre)
- **file** fájl típus lekérdezés (Nemcsak jegyzékre)

Parancsok: másolások, mozgatások

- cp copy, másolás
- mv move, mozgatás (rename helyett is!)
- ln (link) "linkel"
- rm (unlink) "linket" töröl, remove: file törlés

- find keres fájlt egy fán és csinál is valamit (bonyolult, de nagyon hasznos!)

Parancsok: állapotlekérdezések

• ps	processzek listája
• file, ls, pwd	ld. fönn
• date	dátum, idő lekérdezés
• who, w, rwho, rusers	ki van bejelentkezve?
• rup	mely rendszerek élnek?
• top	erőforrás használat csúcsok
• osview, vmstat	erőforrás használat
• last	utolsó bejelentkezések
• uptime	mióta fut a rendszer?

Parancsok: állapotlekérdezések$_2$

- **finger** — **ki kicsoda?**
- **passwd** — **jelszóállítás**
- **chsh, chfn,** — **név, induló burok stb. beállítás**
- **ldapsearch** — **LDAP adatbázis lekérdezés**
- **xhost** — **X11 munka engedélyezése**
- **set** — **környezet (environment) lekérdezése**
- **du, df, quota** — **diszk, fájl használat**

Parancsok: processz indítás, vezérlés

- **sh, csh, ksh, tcsh, bash** — **shell indítás**
- **exec** — **processz indítás**
- **kill** — **processz "megölése", szignálküldés**
- **sleep** — **processz altatása**
- **wait** — **processz várakoztatás**
- **at** — **processz indítása egy adott időpontban**
- **nohup** — **kilépéskor ne ölje meg**
- **test** — **kifejezés tesztelése**

Parancsok: processz indítás, vezérlés$_2$

- expr kifejezés kiértékeltetése
- if, case, for, do while vezérlő szerkezetek
- break, continue vezérlő szerkezetek
- echo argumentumai visszaírása (meglepően hasznos valami)
- mplayer (video lejátszó)
- xmms, aumix (audio lejátszás)

Parancsok: kommunikáció

- ssh, telnet, rlogin, rsh — kapcsolatlétesítés
- rwho, rusers, finger — lásd állapotlekérdezések
- write — üzenet konzolokra vkinek
- talk, xtalk — interaktív "beszélgetés"
- mail, mutt, pine, mozilla-thunderbird — e-mail
- ftp, scp — fájl átvitel
- lynx, w3m, firefox, netscape WWW — böngésző (kliens)

Parancsok: hasznos szűrők

- **grep** mintakereső
- **awk, nawk** mintakereső feldolgozó
- **wc** sor, szó, karakterszámláló
- **sed** áradatszerkesztő
- **cut** mezőkivágó
- **tail, head, more** egyfajta kiírók
- **sort** rendező

Parancsok: tanuljunk

- **man** on-line kézikönyv lap lekérdezés
- **apropos** kézikönyvben kulcsszó (ha van)
- **whereis** hol van egy parancs
- **whatis** man lap leírás
- **xman** X11-es kézikönyv (grafikus)

Ismételjük: a parancs fogalma

- **Fehér karakterekkel határolt szavak sora**
  - **első szó a parancs neve,**
  - **többi szó az argumentumok.**
- **Az sh beolvassa, értelmezi, átalakítja, végrehajtja**
  - **saját maga (belső p.),**
  - **gyermek processzben (külső p.)**

Mindkét esetben van visszatérési értéke!

Vannak szabványos adatfolyamok!

A csővezeték fogalma

- **A csővezeték (pipe) parancsok sora | operátorral összekötve:**
- **parancsbal | parancsjobb**
- **Szemantikája: végrehajtódik a parancsbal, szabványos kimenete egy csőbe képződik, majd végrehajtódik a parancsjobb, aminek szabványos bemenete erre a csőre képződik.**
- **A cső visszatérési értéke: a parancsjobb visszatérési értéke.**
- **A parancs degenerált cső.** **Példa:**

```
> ypcat passwd | grep kovacs
```

A parancslista

- **Csővezetékek sora listaoperátorral összekötve:**

csőbal op csőjobb

Listaoperátorok:

&& || # magasabb precedencia, de alacsonyabb mint a |

& ; \n # alacsonyabb precedencia

A szemantika:

; \n soros végrehajtása a csöveknek

& aszinkron végrehajtás (csőbal háttérben)

&& folytatja a listát, ha csőbal normális visszatérésű

|| folytatja a listát, ha a csőbal nem normál visszatérésű

Parancslisták

- **A lista visszatérési értéke az utolsó cső visszatérési értéke.**
- **Háttérben futó cső visszatérési értéke különlegesen kezelhető.**
- **A cső degenerált lista (ahol ezentúl listát írunk, írhatunk csövet, sőt parancsot is!)**
- **A && és || operátoros listáknál először láthatjuk a visszatérési érték értelmét! Valóban a vezérlés menetét befolyásoljuk!**

Példák

> cd ide && rm junk # csak akkor töröl, ha ...

> ls ide || cp valami ide # ha nincs ide, készíti

Nézzük, magyarázzuk ezt!

> (mv a tmp && mv b a) && mv tmp b

Példák

- „Háttérben” futtatunk

```
> myprog 1 2 &
125                                   <- PID!
>
```

- Mi a különbség?

```
> echo valami echo valami
valami echo valami
```

```
> echo valami; echo valami
valami
valami
```

Az adatfolyamok átirányítása

- **Mielőtt a lista/parancs végrehajtódik, az sh nézi, van-e átirányító operátor > >> < a szavakban (szavak előtt). (A << különleges!)**
- **Ha ilyeneket talál, szeparált processz(eke)t készít, azokban az adatfolyamokat fájlokba(ból) képzik le, majd abban hajtják végre a listát/parancsot. (Csőnél is szeparált processz!)**
- **A szeparált processz(ek)nek átadja a “maradék“ argumentumokat.**

Az átirányító operátorok

< file # file legyen az stdin
> file # file legyen az stdout, rewrite
>> file # file legyen az stdout, append
<<[-]eddig # here document, beágyazott input

- Példa:

> mypr < innen > ide elso masodik
0 1 2

> exec >outfile 2>errorfile # szkriptben használatos
az átirányítások a kurrens shellre hatásosak
az átirányítás OK, visszatérési érték 0 (normális)
ha átirányítási hiba van, a visszatérés érték 1

Az átirányító operátorok

<<[-]eddig # here document, beágyazott input
Példa: (a ! jelzi az adatsorok végét)
a.script
----
grep ezt <<!
elso sorban van ezt
2. sor, ebben nincs
3. sor
!
echo ' na mi van?'

Így indíthatjuk, és az alábbi az eredmény:
$ a.script
elso sorban van ezt
na mi van?
$

Fájlnév kifejtés (behelyettesítés)

- **Argumentumokban használt metakaraktereket (közöttük a dzsókereket: * ? [ ]) a burok a lista/parancs végrehajtása előtt különlegesen kezeli.**
- **Ha a szavakban dzsókereket talál, azt a szót *mintának* (pattern) veszi.**
- **A minta behelyettesítődik alfabetikus sorrendű fájlnevek listájává, olyan nevekre, melyek a fájlnév-térben illeszkednek a mintára.**
- **Csak ezután hajtódik végre a parancs/lista.**

Az illeszkedés

- „Szokásos” karakter önmagára illeszkedik …
- A ? egyetlen, bármely karakterre illeszkedik.
- A * tetszőleges számú, tetszőleges karakterre illeszkedik.
- A [...] illeszkedik egyetlen, valamelyik bezárt karakterre.
- A [!...] illeszkedik egyetlen, bármely, kivéve a ! utáni karakterre.
- stb., nézz utána!

Példák

- **Tegyük fel, az aktuális jegyzékben van 4 fájl:**

a abc abc.d xyz

> ls * # ⇒ ls a abc abc.d xyz

> ls a* # ⇒ ls a abc abc.d

> ls [a]?? # ⇒ ls abc

> ls [!a]?? # ⇒ ls xyz

Vegyük észre, előbb megtörténik a behelyettesítés, csak azután hajtódik végre az ls parancs!

Vö: > rm * és DOS> DEL * Vigyázz: > echo *

a abc abc.d xyz

Változó definíció és behelyettesítés

- Vannak a buroknak definiált változói (és lexikális konstansai)
- Mi is definiálhatunk változókat (és ezeket érvényességi tartományukon belül használhatjuk)
- A definíció: valtozo=szöveglánc
- A behelyettesítés: $valtozo
- Pl.

  var=pipara

  echo ‘ragyujtottam a ‘$var

A metakarakterek semlegesítése

- A metakaraktereket (operátorok, elválasztók, dzsókerek, átirányítók stb.) szükség esetén quota-zhatjuk
- Egyetlen karakter semlegesítése: \spec_karakter
- Több karakter semlegesítése:

  ‘szöveglanc’

  ”szöveglanc” # a változó behelyettesítés marad
- Pl.

  echo ”ragyujtottam a $var”

A metakarakterek semlegesítése

- Pl.

```
# a - sh változó, $a - kifejtése
$ a=abc                 # értéket kap az a
$ echo  '$a'            # semlegesítve a $ kifejtő operátor
$a
$ echo  "$a"            # hatásos a $ operátor
abc
$ echo  "$\a"           # a \a-val az a karaktert értjük
$\a
$
```

A burok

- A burok (shell) kifejezés kettős értelméből mi csak az egyiket vettük: a shell parancsértelmező processz
- Másik értelmét (a burok egy programnyelv is), később
- A metakarakterek semlegesítése, további „behelyettesítések”, a burok adatszerkezetei stb. későbbi témák.
- Lássuk be, a parancsnyelvi felhasználói kezelő hatékony, bár kényelmetlen.

A motiváció a továbblépésre

- Hogyan lehetne barátságosabbá tenni a kezelést (a számítógéprendszerrel való kapcsolattartást)?
  - Heterogén tudású felhasználók, eltérőek az igények
    - Kezdő felhasználók igényei
      - kevés, egyszerű parancs legyen,
      - biztonságosak legyenek (ne lehessen nagy kárt okozni),
      - részletes és környezetfüggő segítség,
      - akár a teljesítmény rovására is.
  - Voltak alfanumerikus menüs kezelők …
    - szöveges menükből választani egyszerűbb ...
  - A nehezen megjegyezhető parancsok helyett
    - ikonok, menüelemek kiválasztása,
    - grafika!

Grafikus felhasználói felületek

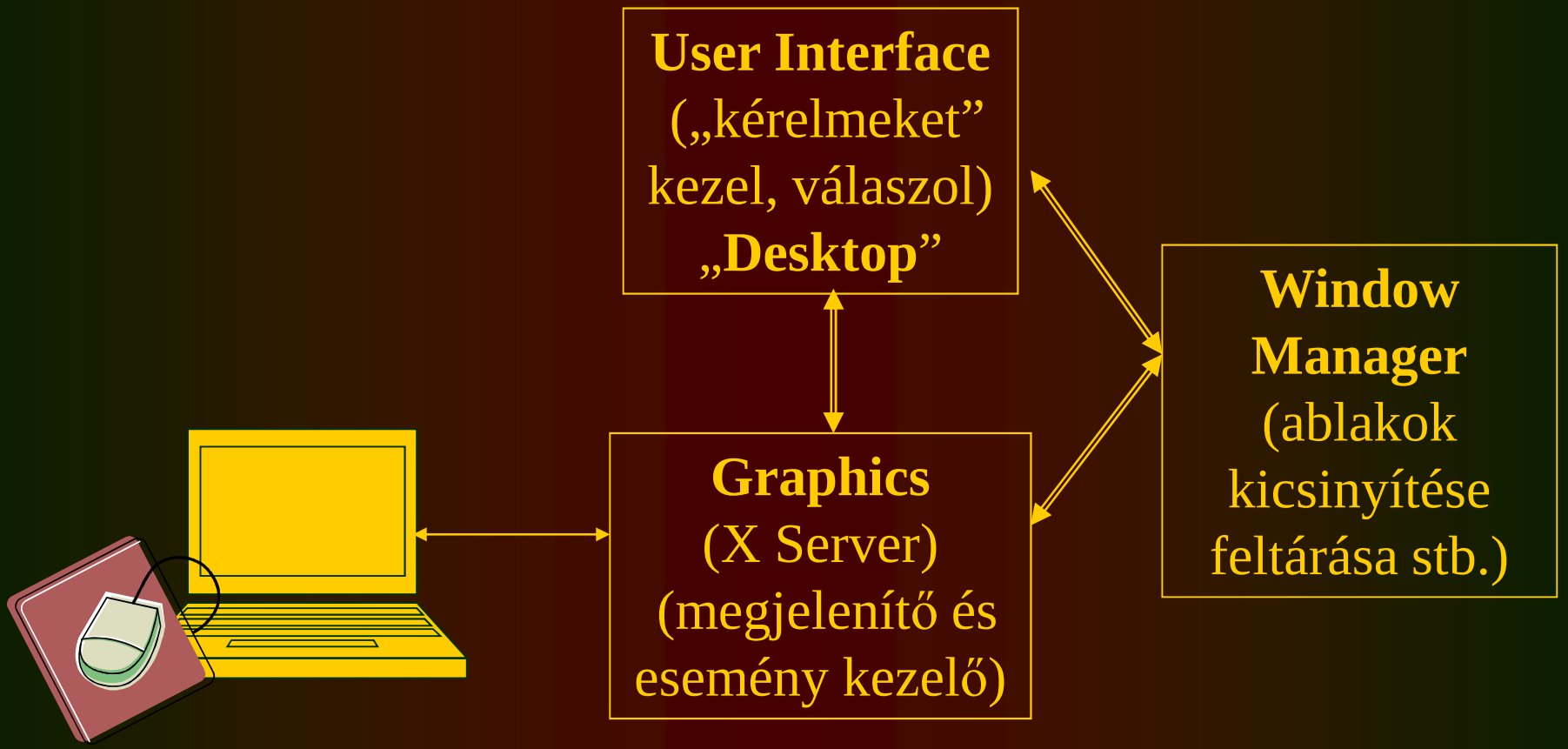
- **Alapgondolat: vagy egy *munkaasztalunk* (desktop)**
- **rajta:**
  - **szerszámos polc (fiók),**
  - **iratok (dokumentumok),**
  - **irattartók (mappák), bennük iratok, irattartók;**
  - **eszközök (szemetesláda, irat-daráló) stb.**
- **Ez jelenik meg stilizáltan.**
- **Parancsnyelv: kijelölés, kattintás, kettős kattintás, vonszolás, szövegbegépelés stb.**
- **Válasznyelv: ikonok, ablakok, menük stb.**

Ki volt az első?

- Vita az elsőségért: Xerox versus Apple
  - Xerox PARC: Smalltalk prototype, később ALTO STAR
    - Ezek nem voltak kereskedelmi termékek ...
  - Apple: Macs UI, Lisa
    - Az Apple alkalmazott Xerox PARC kutatókat,
    - de saját GUI-t fejlesztett.
    - Ez volt az első kereskedelmi forgalomba is hozott GUI.
- Apple versus Microsoft
  - Kezdetben az MS az Apple alkalmazásírója (kapott engedélyt a Macs UI használatra) és az IBM rendszerszoftver írója volt …
  - Mikor összeveszett az IBM-mel, kiadta a Windows 1.0-át
    - ez az MSDOS fölötti GUI, "lopott" elemekkel … Pert vesztett az MS

A modell

- **Rendszerint több processz, különböző funkciókkal**
  Display/Session Manager – X Server-bejelentkezés-WM-Desktop



CLI és GUI összevetés

- **Mi kell CLI-hez?**
  - **Kapcsolatépítő (ssh – init - tty)**
  - **Ülés létesítő (ssh – login)**
  - **CLI (ssh – bash)**
- **Mi kell a GUI.hoz?**
  - **Kapcsolat és ülés építő: Display Session Manager**
  - **Window Manager (ikonizáláshoz, ablakmozgatáshoz)**
  - **GUI (Desktop és X server/munkahelykezelő)**

Egy tipikus X-es munkaállomás

- Rendszerindításkor indul a Display/Session Manager.
- Ez indítja a gépen az X-szerver-t is.
- A Display/Session Manager adja a “bejelentkezési“ ablakokat (login-password), segíti az ellenőrzött ülés létesítést.
- Sikeres ülés létesítés esetén indít
  - Window Manager-t,
  - Desktop-ot. Utóbbival “kezeljük“ a gépet.

A Windows Desktop-ja

- **Objektumok: egységesen kezelt entitások**
  - **vannak tulajdonságaik (függenek az objektumtól),**
  - **műveletek végezhetők rajtuk.**
- **Pl. merevlemez objektum,**
  - **tulajdonsága: kapacitása, szabad kapacitása stb.**
  - **műveletek: formázás, hibaellenőrzés stb.**
- **Pl. fájl objektum**
  - **tulajdonsága : neve, méret, típusa, létesítés dátuma stb.**
  - **műveletek : másolás, törlés, átnevezés stb.**

A Windows Desktop-ja

- **Objektumok hierarchiája:**
- **ha egy objektum más objektumot tartalmaz, akkor “mappa” (folder, irattartó).**
- **Minden directory mappa, de nem minden mappa directory!**
- **A legfőbb mappa maga a munkaasztal (desktop). Az egész képernyő.**
- **A munkaasztal legfőbb elemei:**

– Start gomb, **– Tálca (TaskBar),** **– (Parancs) ikonok**

A Windows Desktop-ja

- **Parancsindításnak változatos formái lehetnek**
  - **kettős kattintás bármelyik mappában lévő ikonjára,**
  - **fájl-ikonra, amihez asszociált a kezelő program, kettős kattintás (Pl. doc-ra indul a Word),**
  - **fájl-ikon vonszolása alkalmazás ikonba (pl. doc fájl a Word-be),**
  - **Start gomb almenüjeiből kattintással,**
  - **Start gomb Run menüjével,**
  - **Alkalmazásból az OLE (Object Linking and Embedding: objektumcsatolás, beágyazás) segítségével (Pl. Word dokumentumba ágyazott Excel táblára kattintással)**

Adatátvitel alkalmazások között

- Az említett OLE is (esetleg link jelleggel is),
- Vágólapon át.

- Egyes alkalmazáson belül vonszolással is (copy, move, esetleg link jelleggel)

A CDE

- Common Desktop Environment
  - Integrált, szabványos, konzisztens, konfigurálható,
  - nyílt rendszer elvnek megfelelő, platform-semleges kezelő/felhasználói felület.
  - Grafikus desktop-ok segítségével kezelhetünk.
- A CDE projekt résztvevői
  - beadták tudásuk és technológiai elgondolásaik legjavát,
  - kialakítottak egy funkcionalitás készletet (óriási vita volt a készlet körül)
  - ezt minden (sok) platformon implementálták.
- X11-es, MOTIF

A CDE elemei

- A munkaasztal (desktop), ami tartalmazza
  - A Front Panel-t
    - ikonok és menük készlete, gyors indításokhoz.
  - Szabványos fájl-menedzsert
    - direkt manipulációk az eszköz és fájlrendszereken
  - Alkalmazás-menedzsert
    - A Front Panelből indíthatóan, kimondottan alkalmazások (installált végrehajtható programcsoportok) kezelésére.
  - Többszörös munkaterületeket (Virtual Workspace)
    - Különböző munkákhoz (pl. szokásos irodai tevékenységhez, egy projekt munkáihoz, játéhoz, szórakozáshoz stb.) egy ablakon belül környezetet biztosítanak.
  - Hasznos "szerszámokat"
    - pl. levelező, naptár, kalkulátor, editor, terminál emulátor, segítő, ikon-szerkesztő, stílus menedzser stb.

CDE jellemzők

- **Kényelmes kezelés**
  - **OO jellemzők (eszköz/fájltípushoz rendeltek a funkcionalitások);**
  - **Változatos adatát(be)viteli lehetőségek alkalmazáson belül és alkalmazások között is**
    - **moving, copying, linking, creating, deleting, sharing jellegekkel**
    - **direkt átvitel (vonszolás), vagy több "fokozatban"**
    - **indirekt átvitel (primary transzfer: kijelöl+céloz+TRANSFER; quick transzfer bevitelre; clipboard transzfer)**
  - **ToolTalk protokoll**
    - **Különben független (függetlenül fejlesztett) alkalmazások közötti üzenetváltási szolgáltatás**
    - **"Becsomagolt" objektumok adódhatnak át**

CDE jellemzők

- **Kifinomult viszonymenedzser**
  - **Legfontosabb tulajdonsága:**
    - **kilépéskor megőrződhet a programok futási állapota,**
    - **belépéskor folytatható a munka.**
    - **Lehet alapértelmezési (default) állapottal is indulni.**
- **Igen gazdag az eszközkészlet**
  - **Sun és Novell adalékok többségükben, de mindenki mindent beadott.**
  - **Szinte minden, ami kellhet egy átlagos felhasználónak**
    - **mail, calc, szövegszerk., fájlkezelő, nyomtatókezelő, környezet-érzékeny súgó, stíluskezelő (háttár, színek stb. beállítás);**
    - **pl. személyenkénti naptárak, de "egymásra vetíthetők": közös szabad időpont kiválasztására.**

CDE jellemzők

- **Alkalmazás-fejlesztő komponensek**
  - **Motif stílusú alkalmazásfejlesztéshez grafikus készlet, amiből építkezhetünk (elsősorban a GUI-t állítjuk össze), és**
  - **generálódik a C forrás kód (ezt továbbszerkesztve, kiegészítve az alkalmazási logika szerinti kódokkal, fordítva linkelve gyorsan kész az alkalmazás)**

Terjed? Nem terjed?

- Az egyes rendszerszállítóknak van saját GUI-juk is
- 1996-ban azt gondolhattuk, roppant népszerű lesz.
- Mi volt az oka a lassú terjedésnek?
- Magánvéleményem: a WWW böngészők az ok!
  - Ezek is sokféle funkcionalitást adnak, “egységes“ felületen.
  - a WEB robbanás leállította a CDE terjedést.
- Mai szállítók CDE-vel szállítanak
  - AIX, Solaris, HP-UX, Digital Unix, Linuxok

Számítógép architektúrák

Felhasználói felületek

Vége