Régi és új felhasználói felületek

– házi dolgozat a Számítógépes kultúra c. tárgy keretében –

írta: Szabó Péter (MVBYN6), IV. éves műszaki informatikus hallgató
dátum: 2003. június 3.
a tárgy előadója: Dr. Galántai Zoltán

1. Áttekintés

A kezdeti számítógépeknek nem volt felhasználói felületük (user interface, user environment), ugyanis a felhasználó nem tudott párbeszédet folytatni a géppel. A beviteli eszköz a lyukkártya volt, melybe a betűket egy speciális írógéppel kellett lyukasztani. A lyukkártyákból egy program állt össze, melyet a felhasználó átnyújtott az operátornak, majd várt néhány napot, és a program futásának eredményét megkapta leporellóra nyomtatva. Ha akárcsak egy zárójelhiba is volt a programban, az nem futott le, a hibát a megfelelő lyukkártya lecserélésével lehetett javítani, és ismét napokat kellett várni az eredményre. Az operátor dolga volt a lyukkártyákat megfelelő sorrendben behelyezni az olvasóba, a lyukkártyát értelmező programot tartalmazó szalagot betöltetni a géppel, és a futási eredményt a megfelelő felhasználóhoz eljuttatni. Látható, hogy akkoriban a számítógép-használat egyenlő volt a (főleg tudományos céllal kutató) programozással, és hogy operátori segítség nélkül nem is lehetett dolgozni a géppel.

A később megjelenő terminálok segítségével a felhasználó már közvetlenül a gépet szólította meg, és már nem csak számításra használta, hanem pl. adattárolásra és -visszakeresésre és elektronikus levelezésre. Ahogy a technológia egyre kisebb és gyorsabb eszközöket tudott gyártani, a felhasználók többsége megengedhetett magának egy saját, személyi számítógépet (personal computer, PC), nem volt szükség több terminálra, ezért a monitor és a billentyűzet már a számítógép tartozéka lett. Új beviteli eszközként megjelent az egér. A megjelenítő-technológia és a memóriák fejlődésével az 1990-es években általánossá és általánosan elvárttá vált a grafikus felület. Nem mondható el azonban, hogy az új technológiák kiszorították a régiek köré kapcsolódó kultúrát és felhasználói szokásokat: a kötegelt feldolgozás, a szöveges és a grafikus felhasználói felület egyszerre van jelen a modern számítógépekben, s bár kétségkívül a grafikus felület felhasználói köre a legszélesebb, mindegyik felületet érdemes megismerni, hogy mindig az adott feladathoz legalkalmasabbat választhassuk.

2. Szöveges terminálok

A terminál fogalma egyidős az időosztásos rendszerekkel. Egy időosztásos (time division, multitask) rendszer megosztja a figyelmét és számítókapacitását több program között, sűrűn (másodpercenként akár többezerszer) átkapcsol az egyik program futtatásáról a másikéra, ezért úgy tűnik, mintha ezek a programok párhuzamosan futnának. Az időosztásos rendszerhez már készíthető felhasználói felület, a terminál, melyet a felhasználó jegyzetfüzetként és írógépként használva valós időben fogalmazza meg a gép számára kiadott parancsokat (command). A megfogalmazás alatt a felhasználó maga kommunikál a géppel, a terminál kijelzőjén látja a már részlegesen beírt parancsot, lehetősége van javításra és a parancs befejezésére. Ez az interaktív parancsbevitel elenyésző részét köti le a gép számítókapacitásának, ezalatt időosztásos módon lehetőség van más felhasználók már elkészült parancsainak futtatására. Ha több parancs is elkészült, akkor lehetséges azokat egyidejűleg futtatni, de ekkor a végrehajtás sebessége csökken. Egy számítógépet tipikusan $10 \ldots 20$ felhasználó érhet el, egy-egy terminál segítségével.

A terminál az írógépes párbeszéd ötletén alapul. Képzeljünk el egy írógépet, melyre a felhasználó és a számítógép felváltva írnak be sorokat. A felhasználó a billentyűzeten begépel egy sort, a számítógép ezt parancsként értelmezi, végrehajtja, és az eredményt a papír következő sorába írja az írógép kalapácsainak elektronikus vezérlésével. A parancsok nagy része az utókor szempontjából érdektelen (pl. a felhasználó megkérdezi a pontos időt vagy rendszerezi a saját file-jait), és sok válasz túl hosszú lehet, ezért anyag- és időpazarlás őket papírra vetni. Ehelyett az írógép kimeneti eszközét, a papírt egy képernyőre cserélték, amely általában fekete alapon zöld színben képes volt betűk megjelenítésére. Így született a (szöveges) terminál, ami egy billentyűzet és egy szöveges kijelző kombinációja. Hátránya a papírhoz képest, hogy magassága kisebb ($25 < 60$ sor), a betűk nehezebben olvashatók, továbbá nehéz „visszalapozni” egy korábbi oldalra. Ez utóbbi problémára több megoldást is látni fogunk.

A terminálhálózatok és az első UNIX rendszerek jellemző felhasználói felülete a soronkénti parancsbevitel volt. A felhasználó a terminál billentyűzetén begépelt egy parancsot, majd az ⟨Enter⟩ billentyű lenyomásával a terminálhoz soros vonalon (RS-232) hozzákapcsolt számítógépre bízta annak végrehajtását. Az eredmény megjelent a szöveges kijelzőn, melynek tipikus felbontása 80×25 karakter volt, tehát a betűk állandó szélességűek (átalában legfeljebb 8 képpont) és magasságúak ($12 \ldots 16$ képpont) voltak. A túl hosszú sorok a következő sorban folytatódtak, és a túl hosszú kimenet eleje felül kigördült (scroll), ezzel alul helyet biztosítva a végének. Egy ideig olyan lassúak voltak a számítógépek, hogy a felhasználó görgetés közben végig tudta olvasni a kimenetet. Később a terminálokat felszerelték megállító funkcióval (flow control), ami billentyűlenyomásra oldalanként görgetett, így a felhasználó annyi időt szánhatott egy oldal elolvasására, amennyire szüksége volt.

Gépeléskor a félig kész parancs aktuális állapota a kijelzőn volt látható. Hiba esetén a ⟨BackSpace⟩ billentyűvel lehetett az utolsó karaktert törölni (de a nyílbillentyűkkel nem lehetett korábbi karakterekhez visszamenni), vagy Ctrl-⟨C⟩-vel az egész parancsot. Ezt az nehézkes és rugalmatlan beviteli módot folyamatosan fejlesztették, mivel buta (dumb) terminálok esetén az új beviteli és szerkesztési funkciókat a számítógép szoftverének frissítésével lehetett hozzáadni, nem kellett magát a terminált lecserélni. A buta terminálok billentyűleütéseket (és nem kész parancssorokat) továbbítottak a számítógépnek, amely válaszul a terminálra karaktereket rajzolt. Ez a karakterenkénti késleltetés fölöslegesen terhelte mind a számítógépeket, mind a hálózatot, és a felhasználók hatékony munkavégzését is zavarta. Megoldásként megszülettek a terminálokba épített előfeldolgozó egységek (front-end processor), melyek a felhasználó közvetlen közelében bonyolították az interaktív parancsbevitelt, sőt, még az is előfordult, hogy a beviendő BASIC program tokenizálását is átvállalták a számítógéptől, és így a szintaktikai hibát azonnal tudták jelezni. Az előfeldolgozás mégsem vált népszerűvé, mert szoftverfrissítés körét kiterjesztette az egyébként is nehezen elérhető, és kevéssé flexibilis terminálokra. A jellemzően interaktív programok is sororientáltak voltak. Például az egyik tipikus szövegszerkesztőben (ed) a felhasználó kilistázta a file számára érdekes részét, a lista alapján megadta a módosítandó sor számát, majd az egész új sort be kellett gépelnie, beleértve a változatlan részeket is.

Megjegyezzük, hogy a mai telnet és SSH protokollok kliensei is buta terminálként viselkednek. A késleltetés például modemes internetelérésnél szembetűnő, amikor minden begépelt betű megjelenésére több tizedmásodpercet kell várni, és különösen zavaró, hogy a nyílbillentyűk lenyomásának idejét a késleltetés-ingadozás miatt felhasználó nem tudja megbecsülni, ezért a kurzor távolra mozgatása esetén gyakran tetemes utólagos korrekcióra van szükség. A problémát csak tetézi, hogy a nyílbillentyűk alapból három karaktert küldenek, tehát lassabban vihetők át, mint a sima betűk.

A UNIX-ok mai szöveges felhasználói felülete a terminálhálózatokból alakult ki mind a terminálok, mind az interaktív programok fejlődésével. A terminálok működése gyorsult, manapság az információmegjelenítés sebessége jóval nagyobb, mint ahogy azt az ember be tudja fogadni; továbbá megjelent a kurzorpozícionálás lehetősége. A kurzor egy villogó vonal vagy téglalap, amely a kijelzőn a következő megjelenítendő karakter helyét mutatja. A gyors, de buta terminál elküldte a billentyűleütést a számítógépnek, amely eldöntötte, hogy ennek hatására a képernyő mely részét és hogyan kell megváltoztatni, majd kurzorpozícionáló escape-szekvenciákat[1] küldött vissza a terminálnak, ezután felülíratta a szóban forgó részt, és a kurzort az új helyére pozícionálta. Megjelentek a színes terminálok is, melyek 16 féle színben (az RGB (piros, zöld, kék) alapszínek 8 db kombinációja, és ugyanezek világosabb változata) voltak képesek egy karaktert és ettől függetlenül annak hátterét megjeleníteni. A színváltoztatás is escape-szekvenciákkal történt. Manapság az egér alaptartozéknak számít, ezért a terminálként üzemelő PC-k az egéreseményeket (mozgatás, kattintás) is képesek escape-szekvenciaként elküldeni[2].

Számos program nem használja ki a terminálok interaktív lehetőségeit, hanem továbbra is megtörő és kigördülő sorokat produkál a képernyőn, és sororientált, ⟨BackSpace⟩-szel javítható bemenetet vár. Az ilyen programok kimenete egy UNIX csővezetéken (pipe) egy listázóprogramnak továbbítható (pl. less), amely interktív módon kétirányú görgetést és keresést is lehetővé tesz. Sőt, bizonyos terminálok rendelkeznek scrollback buffer-rel, melyben a nem túl régen kigördült szöveget tekintheti meg a felhasználó. Ezek a módszerek tökéletesen lehetővé teszik, hogy a hagyományos, szöveges programok

[1] azért nevezzük őket így, mert a 27-es kódú ASCII *escape* karakterrel kezdődnek

[2] A Linux konzol gpm egérkezelő-megoldása negatív példaként említhető, mert a szekvenciák helyett egy rendszerspecifikus kerülőutat választ, amely például SSH-n sem megy át. Jól működik viszont az XTerm emulátor.

kimenetét a felhasználó a saját ritmusában, akár vissza-visszatekintve nézze meg.

A 21. században ódivatúnak tűnhet az ilyen puritán, nem interaktív programok használata, de ezek óriási előnye, hogy a csővezetéken más programokhoz csatlakoztathatók, és így egyszerű, szabványos kockákból gyorsan építhetők feladatspecifikus script-ek. Ebből a szempontból kifejezett előny, ha egy program futása során nem tesz fel váratlan kérdéseket. Az ilyen programok használatának megtanulása hosszabb, mint a grafikus alkalmazásoké, melyekben csak néhány pipát kell elhelyezni, vagy jó helyen kettőt kattintani. Jelen sorok írója azt figyelte meg, hogy a nem interaktív programok felhasználói jobban értik, hogy mit miért csinálnak, és sosem végzik el ugyanazt a mozdulatsort kétszer, mert a hasznosnak bizonyult parancsokat feljegyzik vagy script-et építenek belőle későbbi felhasználás céljára. Ily módon a munka hatékonysága – különösen adatmanipuláció és szoftverfejlesztés esetén – épp attól lesz nagy, hogy a legtöbb program *nem* interaktív.

A leggyakoribb interaktív szöveges programok a következők: a shell (parancsértelmező, burok, héj), a szövegszerkesztő (text editor) és -megjelenítő, az interaktív filekezelő, folyamatirányító programok (pl. ipari vezérlő, zenelejátszó), webböngészők. Ezek sokat fejlődtek a sororientált bevitel óta. A shell egysoros parancsok szerkeszésére, más programok indítására, és egyszerű programozásra (shell script-ek) alkalmas. Fontos interaktív lehetőségek a kurzormozgatás, a history (korábban kiadott parancsok visszahívása), a kiegészítés (⟨Tab⟩-ra megjeleníti az érvényes parancsok vagy a lemenzen levő file-ok listáját), a filenevek illesztése joker karakterek alapján (wildcard filename globbing), a parancsváró prompt, amelyben megjeleníthető az előző parancs visszatérési kódja, az aktuális könyvtár, a felhasználónév, a gépnév, a pontos idő és sok minden más. A shell viselkedését midenki testreszabhatja saját igényei szerint, saját színeket, billentyűkombinációkat és parancsrövidítéseket (alias) lehet definiálni. Ez hasonlít a mobiltelefonok csengőhang-állítására, azzal a különbséggel, hogy a jól beállított shell a mindennapi munkát észrevehetően gyorsítja.

A Linux szöveges módban alapból 6 db virtuális terminált kínál, melyek között a Ctrl-Alt-⟨F1⟩ ... Ctrl-Alt-⟨F6⟩ kombinációkkal válthatunk. (Nem túl nagy fáradsággal a virtuális terminálok száma feltornázható, de 12 fölé nem érdmes menni, mert elfogynak a funkcióbillentyűk, és a váltás nehézkessé válik.) Érdemes ezek közül sokat kihasználni. Például az egyik terminálon a levelezőprogramot futtathatjuk, a másikban a zenelejátszót, a harmadikban a szövegszerkesztőt, amelyben éppen forráskódot szerkesztünk, a negyedikben a fordítóprogramot (compiler), az ötödikben a lefordított programon végezhetünk próbafuttatásokat, a hatodikban dokumentációt olvashatunk, a hetedikben egy webböngésző futhat, a nyolcadikban pedig egy shell, a kisebb, ad-hoc módon felmerülő parancsok futtatására, a kilencedikben pedig egy file-kezelő. Ha egyik terminálon futó programból sem lépünk ki idő előtt, akkor kihasználhatjuk a jó helyzetet (a filekezelő mindig a megfelelő könyvtárban áll, a webböngésző a jó site-on van stb.), és a history-t is, tehát semmit sem kell kétszer begépelni, elég az előző parancsot előhívni, és esetleg átírni benne néhány karaktert.

Tovább növeli a kényelmünket, hogy az egyik virtuális terminálon megjelenő szöveget ugyanarra, vagy egy másikra az egér segítségével átmásolhatjuk (a másolat billentyűleütések formájában jelentkezik), így például nem kell begépelnünk URL-eket, példaprogramokat, vagy programoknak a dokumentációban leírt mintafuttatásainak parancsorait. Fileneveket és parancsneveket sem kell teljesen begépelni, a ⟨Tab⟩-ot a shell-ben használhatjuk kiegészítésre. Ha egy bonyolultabb, jól működő parancsot sikerül előállítanunk, érdemes azt egérrel a szövegszerkesztőbe másolni, és elmenteni shell script-ként későbbi használatra. Semmit sem kell kétszer begépelni, viszont amit egyszer jól begépelünk, az később is ugyanúgy, visszakérdezés és kattintgatás nélkül működik. A fent leírtakat foglalja össze az angol nyelvű szójáték: „Linux has a CLUE: Command-Line User Environment”. Ez persze nem csak a Linux-ra igaz, hanem a UNIX-filozófiára épülő összes rendszerre, de Linux alatt jól megfigyelhető, hogy a gyakorlott felhasználó és programozó mennyivel produktívabb, mint a grafikus rendszereken nevelkedett társa. Példaként említhetjük, ha egy file-ban az összes `a` betűt `b`-re kell cserélni. Ismert, hogy a `perl -pi -ey/a/b/ FILENÉV` parancs ezt megcsinálja. E tudás birtokában a csere a parancs begépelésével együtt 20 másodperc alatt elvégezhető. Ugyanez grafikus rendszeren az alábbi műveletsort igényli:

1. az intéző (file-kezelő) elindítása a start menüből, és a gyökértől nyitogatós navigálás a file-ig
2. dupla kattintás a file-on, a szövegszerkesztő (text editor) elindítása
3. a cserélő párbeszédablak előkeresése

4. annak ellenőrzése, hogy a csere karakterenként, és nagybetűkre érzékeny módon fög történni
5. az összes csere végrehajtása egy gombra kattintva
6. a file elmentése
7. annak ellenőrzése, hogy a szövegszerkesztő nem vágta szét a hosszú sorokat, nem konvertálta a sorelemeléseket stb., tehát semmi olyat nem csinált, amire nem kértük
8. annak ellenőrzése, hogy a szövegszerkesztő tényleg olyan néven mentette el a file-t, ahogy kértük, nem főzött hozzá `.txt` kiterjesztést, stb.
9. a szövegszerkesztő bezárása

Mi történik, ha `a` és `b` betűk helyett számos betűt le kell cserélni, rekurzívan az összes szövegfile-ban? A UNIX-os, Perl-t használó megoldás nem lesz sokkal hosszabb: `zsh -c 'perl -pi ey/korte/meggy/**/*.txt'`. Miután begépeltük, elmenthetjük egy shell script-be vagy egy alias-ba, hogy legközelebb már csak egy vagy két betűt kelljen gépelni. A grafikus rendszerben erre a feladatra nincs beépített megoldás, a párbeszédablakot file-onként és betűnként külön-külön ki kell töltenünk, indokolatlanul fáradságos munkűval.

Ez egy tipikus és meggyőző példa arra, hogy a szöveges felhasználói felületen a megfelelő tudás birtokában hogyan lehet egy egyszerű feladatot pontosan és időhatékonyan megoldani, és hogy ugyanez grafikus, kattintgatós filozófiával mennyire elkedvetleníti a felhasználót, és kizökkenti a gondolataiból. A példa arra hívja fel a figyelmet, hogy a szöveges, nem interaktív programok napjai nincsenek megszámlálva, mindaddig, amíg a felhasználók hajlandóak megtanulni kezelni őket. Ha a felhasználó egyaránt ismeri a szöveges és a grafikus felületek előnyeit, mindkettőből a legjobbat hozhatja ki. Egy szöveges shell ablak UNIX alatt grafikus felületről is előhozható, ilyen például az XTerm és az RXVT. Ezekből több is kifér egyszerre. Fontos, hogy a kétfajta felület között a különbség filozófiai: Windows-ra is le lehet tölteni az zsh-t és a Perl-t, és Linux-ra is vannak grafikus szövegszerkesztők.

Érdekes, hogy az ablak-koncepció a terminálokon is hamar megjelent, bár az átlapolódásra, az ablakok szabad méretezérése, mozgatására és takarási sorrendjének (z-order) változtatására még sokáig várni kellett. Az ablak a képernyő egy téglalap alakú logikai tartománya, melyet vonalrajzoló karakterek és/vagy eltérő háttérszín különít el a környezetétől. Az ablakban gyermek-ablakok is helyet kaphatnak, ezek nagy részét – funkciójuk alapján – widget-nek (control, vezérlőelem) nevezik. Tipikus widget a legördülő menüsor, a nyomógomb, az egysoros és többsoros szövegbeviteli mező és a felirat. Az interaktív programok szerzői általában törekednek a jó helykihasználásra, kerülik a grafikus világban megfigyelhető pazarlást. Az ablakok kerete gyakran hiányzik, a menüsor csak akkor jelenik meg, ha előhívjuk stb. Közös jellemzője még a szöveges interaktív programoknak, hogy használatuk megkezdése előtt érdemes a leírásukat végigolvasni, hogy kiderüljön, melyik billentyű mit csinál. Az egérnek általában másodlagos szerepe van, mert csak később jelent meg.

A korai idők két népszerű szövegszerkesztője a vi és az Emacs. Ezek mellé az évtizedek során számos egyéb program felsorakozott, melyek a modern igényeket, köztük a buta, feledékeny és együgyű felhasználókat is kiszolgolják. E sorok írójának kedvenc szövegszerkesztője a joe, amely lassú modemes kapcsolat esetén is gyors válaszidőt biztosít, nem túlméretes, billentyűkombinációi testreszabhatóak, nem rontja el a bináris file-okat, több file-t kezel egyszerre, használata során nem kell egérhez nyúlni, képes a jobb margót figyelembe véve újratördelni egy bekezdést, és nem csökken a sebessége hosszú fileok és hosszú sorok esetén. Fontos szempontok még, hogy legyen használható súgó, a menüből vagy alap billentyűkombinációval az összes parancs elérhető legyen, a programok forráskódját ki tudja színezni a szintaxis alapján (syntax highlighting), reguláris kifejezésen alapuló kereső és cserélő legyen benne, lehessen benne intervallumot és téglalapot kijelölni, és azokat másolni, a tab méret állítható legyen, a színkiosztás állítható legyen, a betűtípus állítható legyen, a program ne szálljon el, gyorsan induljon el, saját script-nyelvén viselkedése módosítható legyen, legyen olyan módja, ahol a Windows-ban megszokott menürendszer és billentyűkombinációk érvényesek stb. Az Emacs számos külső funkciót is megvalósít, például forrásszintű nyomkövetés (debugging), Usenet hírcsoportok olvasása, webböngészés, GNU Info és UNIX manual page-ek megjelenítése, filekezelés, filemásolás FTP-vel és SCP-vel stb. A szövegszerkesztők melletti kiállás hevessége gyakran a vallási vitákéval vetekszik. Ez egyrészt jó, mert a választékban vannak sok szempontból kiváló szövegszerkesztők, másrészt mindegyiknek vannak olyan hibái vagy furcsaságai, amit a felhasználók egy jelentős rétege nem visel el.

Összességében elmondható, hogy a text editor-ok eredményes futtatása terminálon is megvalósítható, és a legtöbb értékelési szempontot nem befolyásolják a terminálok alábbi korlátai. A mai terminálok egyik nagy hátránya, hogy nem különböztetik meg elég finoman a billentyűkombinációkat, például a Ctrl-⟨Home⟩ és a Shift-⟨Home⟩ ugyanazt küldik, továbbá hogy a szabványosítás nem elég sikeres, számos escape-szekvencia szabvány létezik, de egyik sem elég általános, és az implementációk sem követik őket pontosan. Gyakran, főleg heterogén környezetben való SSH-zás közben előfordul, hogy a képernyő összezavarodik, az egészet újra kell rajzolni. Kétfajta terminációadatbázis is létezik (ahonnan a programok kinyerik, hogy milyen szekvenciákat kell küldeniük), a termcap és az újabb terminfo, ezek néhol egymással sem konzisztensek, de különböző gépeken is különbözőek. Óriási hátrány még, hogy az ⟨Esc⟩ billentyű egyszeri leütését a nem prefix kódolás miatt lehetetlen érzékelni és megkülönböztetni például egy olyan funkcióbillentyű-leütéstől, ami lassan érkezik meg a hálózaton.

Mindezek ellenére elmondható, hogy az escape-szekvenciákon alapuló, terminfo adatbázis alapján megvalósított buta terminálmodell lehetővé teszi különböző típusú hardver terminálleszközök (pl. a több, mint 30 éves VT-52), a PC-n futó szöveges (pl. Linux konzol) és grafikus (pl. XTerm) terminálemulátorok, VMS konzol, kereskedelmi UNIX-ok konzolai és heterogén (de főleg UNIX és VMS) programok együttműködését. Az SSH és telnet programokkal a felhasználó könnyedén átléphet egyik gépről a másikra, anélkül, hogy felállna a székéből. A terminálok lassú, pl. modemes kapcsolat esetén is használhatóak maradnak (bár a komfortérzet sokat csökken), mert takarékosan bánnak a sávszélességgel. Ilyen mértékű nyíltság és együttműködés elképzelhetlen a kereskedelmi, grafikus operációs rendszerek világában: próbáljunk csak egy Macintosh-os alkalmazás ablakát a szomszédos PC-n megjeleníteni. A grafikus világból jellemző kivétel az X Window System és a VNC, de ezek nem univerzálisan elterjedtek, és modemen keresztül túl lassúak, ezért az együttműködés ott igen korlátozott.

3. Grafikus felületek

A grafikus felhasználó felület (graphic user interface, desktop environment, GUI) a Xerox találmánya, ezt vette át az Apple és a Microsoft is saját operációs rendszereibe. Hálózaton átvihető grafikus felületekre példa a UNIX-okon népszerű X Window System (X11, iksz-tizenegy), a VNC (Virtual Network Computing) és a Windows Terminal Server (pl. Citrix Neptun-kliens). A képernyő legkisebb, független része a képpont (pixel)[3], melynek színe egy-egy RGB-hármassal írható le. A képpontokból nem csak változó szélességű betűk, hanem képek és ablakok is építhetők. Egy ablak átalában téglalap alakú, nem áttetsző (tehát solid), a többi ablakoktól függetlenül mozgatható, a széle kilóghat a képernyőről, egyes részeit vagy egészét más ablakok takarhatják, jól meghatározott helye van a takarási sorrendben, rendelkezik dekorációval (pl. címsor, rendszermenü, minimalizáló gomb, maximalizáló gomb, bezárógomb, átméretező keret). Az ablakokban többnyire szöveg, kép és szabványos widget-ek fordul elő.

A grafikus felület eseményvezérelten működik: a bemeneti eszközök állapotváltozásakor és egyéb esetekben is egy esemény generálódik, melyet az őt érdeklő programok megkapnak és lekezelhetnek. A felhasználó egérrel és billentyűzettel irányítja a számítógépet (fényceruza, gamepad és joystick is előfordulhat, de ezek szakterületekhez kötődnek). A billentyűzet billentyűlenyomás, -ismétlés, és -felengedés eseményeket generál, az egér mozgatás, görgetés (scroll), gomblenyomás és gombfelengedés eseményeket. Az egér azon ablakoknak generál(hat) eseményt, amelyek fölött elhalad, a billentyűzetesemények pedig az ún. fókuszban levő ablaknak továbbítódnak. A fókuszban levő ablak címsora más színű szokott lenni, mint a többié, és egy másik ablakra egérrel kattintva általában az új ablak átveszi a fókuszt. Az ablakon belül a felhasználó a vezérlőelemek között ⟨Tab⟩ és Shift-⟨Tab⟩ lenyomásával választhatja ki a fókuszban levőt, ha ezt az alkalmazás szerzője lehetővé tette számára. Fontos esemény még az „előbukkanás” (invalidate, expose), amikor egy ablak eddig takart része újra láthatóvá válik, tehát a tartalmát újra kell rajzolni. Ekkor egy előbukkanás esemény generálódik, amire az ablak tulajdonosa figyel, és válaszul frissíti az ablak tartalmát.

A grafikus felületnek általában nem része a hanglejátszás (és -felvétel), ez a hálózati grafikus rendszerekből teljesen kimaradt, és különböző kiegészítések formájában elérhető (pl. X11 alá az ESD: Enlightened Sound Daemon). A hang kimaradása magyarázható a nagy sávszélességigénnyel, és az

[3] A Commodore-64-es idők már elmúltak, amikor memóriatakarékossági okokból a képpontok színét nem lehetett a szomszédságuktól teljesen függetlenül megváltoztatni.

eltérő szolgáltatásminőség-igénnyel (késleltetés és késleltetés-ingadozás), továbbá azzal, hogy a grafikus felületek megszületésekor még nem voltak elterjedtek a hanglejátszásra képes munkaállomások. A grafikus felület nagyobb számítókapacitást, több videomemóriát (a képernyő aktuális állapotának tárolására), több memóriát (az események és ablak meta-információ tárolására) és nagyobb hálózatsebességet igényel egy szöveges terminálnál. PC és X11 esetén ez igény kb. 10 MB nagyságrendű, tehát amikor az egész gépben összesen csak 8 MB memória volt, akkor a felhasználó igen türelmetlenül várta, amint a gép az ablakok frissítése között hosszasan swap-pelt. A magasabb erőforrásigények közül ma már csak a hálózatsebességre kell figyelni, a többinek a bolti PC-k alapból megfelelnek. A 3 dimenziós inkrementális képszintézis valósidejű működése a számítógépés játékipar számára nélkülözhetetlen. Ezért megjelentek a 3D gyorsítókártyák, melyek többek között lineáris és homogén transzformációt, mélységszámítást és textúrázást tudnak végezni a fő processzor terhelése nélkül. Ezek többnyire Windows-specifikus megoldások, mert ebből a körből kerülnek ki a vásárlóik. A továbbiakban a hangot, a 3D-gyorsítást és a speciális beviteli eszközöket félretéve arra koncentrálunk, hogy milyen tényezők befolyásolják egy grafikus felület használhatóságát.

Fejlesztési szempontból a grafikus felületet használó szoftverek így csoportosíthatók: szerver, widget-készlet-könyvtár (widget set), ablakkezelő (window manager), kliens felhasználói program, egyéb, a felülethez kötődő alkalmazás. Ha az ablakkezelőt, a widget-készletet és az egyéb, felületspecifikus alkalmazásokat egységes kinézetűre és egymással jól együttműködőre, tehát integráltra fejlesztik, akkor ezt a hármast együtt desktop environment-nek nevezzük. Ilyen desktop environment például a Gnome (sawfish-gnome, GTK, Gnome panel stb.), a KDE (KDE ablakkezelő, Qt, KDE Control Center stb.) és a CDE. Windows és MacOS esetén a desktop envinronment részei csak igen bajosan cserélgethetők egymástól függetlenül, így egyben beszélünk rüluk. Az X11 felhasználóinak nagy része nem kedveli a desktop environment-eket, mert azok nagyok és lassúak, lassan töltődnek be, és a Windows-zal és a MacOS-sel ellentétben nem váltják be az integráltság ígéretét: mindig lesz olyan konfigurációs feladat, amit nem lehet a vezérlőpulton elvégezni, és mindig van olyan program, ami nem található meg a start menüben. Továbbá a súgó sem megfelelő, és bizonyos alapvető dolgok egyszerűen nem működnek a desktop environment-ből beállítva (például bizonyos esetekben hiába kérünk magyar billentyűzetkiosztást, az ő és ű betűk helyett kérdőjelek szúródnak be). Az ilyen, X11 fölötti desktop environment-ek tehát sok esetben segítik a felhasználót, de nem teljesen urai a helyzetnek, és bizonyos problémákat csak a bitekig visszanyúlva, a kattintgatás mellőzésével tud egy hozzáértő megoldani.

A kliens és a szerver a felület hálózaton való átvitelekor különül el élesen egymástól. A képernyő, a billentyűzet és az egér a szervert futtató géphez csatlakozik, az események (beleértve az előbukkanást is) a szerveren generálódnak. Az alkalmazás logikáját pedig a kliens valósítja meg, ő válaszol az eseményekre, és elküldi a szervernek, hogy a mely ablakba mit kell rajzolnia. A rajzparancsok X11 esetén logikaiak, vektorosak (például húzz vonalat, írj ki szöveget, rajzolj ellipszist), VNC esetén mindenképpen pixelesek, ZIP-pel tömörítve. Az X11 előnye, hogy kevesebb komminikációt igényel, ezért gyorsabb, hátránya viszont, hogy extra állapotot tárol a szerveren, továbbá, hogy a kliens nem lehet *teljesen* biztos benne, milyen képpontok jelennek meg a képernyőn, különösen azért nem, mert a megjelenített szöveg a szerverre telepített betűtípusokat használja. A bizonytalanságot a kliens kiküszöbölheti, ha a VNC mintájára mindent képpontonként visz át.

Hasonló probléma a nyomtatásnál is előkerül: általános tapasztalat, hogy a nyomtató a specifikált sebességet csak Windows alatt tartja, Linux alatt súlyos másodperceket gondolkozik minden lap előtt, anélkül, hogy bármit nyomtatna. Ennek az az oka, hogy a Windows-os printer driver-ek többsége vektorosan, logikailag küldi el a nyomtatandó képet, míg a Linux-on elterjedt Ghostscript mindent pixelesen küld, és ezt a nagyfelbontású képpontfolyamot a nyomtatónak több idejébe kerül, amíg fogadja és dekódolja. A Linux-os lassú megoldás előnye, hogy nem fordulhat elő, hogy egy oldal túl bonyolult, ezért a nyomtató feladja, továbbá szoftverhiba és a nyomtató beépített, hiányos fontjai miatt nem kell a kalapos ű-kön búsulnunk.

Az ablakkezelő egy speciális kliens, amely a felső szintű (toplevel) ablakok mozgatásáért, átméretezéséért felel, ő rajzolja ki a címsort, az ablakkereteket, és a tálcát, és gyakran a háttérkép karbantartása is az ő feladatkörébe tartozik. Általában egy grafikus felülethez egyféle ablakkezelőt kínálnak testreszabási lehetőséggel. Nevezetes kivétel az X11, ahol a felhasználók a bőség zavarában válogathatnak. Ismert ablakkezelők: twm, fvwm2, icewm, sawfish, a KDE ablakkezelője, a CDE ablakkezelője, af-

terstep, enlightenment, windowmaker, larswm, xfce, fluxbox, fvwm2-95, 9wm, aewm, blackbox, ctwm, flwm, gwm, ion, mwm, lwm, matchbox, olvwm, oroborus, phluid, pwm, qvwm, ratpoison, sapphire, ude, vtwm, wm2. A UNIX-fejlesztők végre kiélhették kreativításukat, sokan megalkották a saját keretrajzoló és ablakmozgató programjukat. A twm készült el legkorábban, de ebben ki is merül minden érdeme: ronda, nem konfigurálható, és az alapfunkciókat is kényelmetlenül látja el.

Az fvwm2 rendkívül jól konfigurálható, szinte már programozható, sőt, a felhasználó tetszőleges programozási nyelven, jól definiált interfészen csatlakozó modulok írásába vághatja a fejszéjét. Az fvwm2 a hacker-ek csemegéje, bár alapból nem túl tetszetős, és egy ideig el kell bíbelődni a konfigurációs file-lal, mindent kedvünkre beállíthatunk benne. A szolgáltatások közül csak a futó programokat egyetlen sorban kilistázó tálca hiányzik, ehelyett az fvwm2 az egérrel vagy billentyűvel előhívható, többsoros ablaklistát ajánlja, ami mellesleg praktikus is. Az icewm egyszerű, kis erőforrásigényű (a legtöbb ablakkezelő egyébként ilyennek hirdeti magát), és konfigurálás nélkül is egy Windows-hoz hasonló start menüt és tálcát nyújt, így még az egyszerű felhasználó is hamar eligaodik rajta. Sajnos kevés jellemzőjét lehet átalakítani (például billentyűzetkiosztás-váltó ikont nem tudunk a tálcára helyezni), szinte minden bele van drótozva. Csak annyit tud, amennyi fél perc használat után látszik belőle, de azt elegánsan és gyorsan. Létezik hozzá egy iceconf nevű eszköz, melyben háttérképet adhatunk meg, a gyorsbillentyűket átdefiniálhatjuk stb. A különböző desktop environment-ek (pl. KDE és CDE) saját ablakkezelői illeszkednek az egyszerű, buta felhasználó igényeihez: szépek, a vezérlőpulton kattintással konfigurálhatóak, de csak az alapfeladatot tudják, a hatékony használhatóságra kevés figyelmet fordítva (például sok esetben az ablak címsorát nem lehet a képernyőn kívülre (fölfelé) mozgatni). Vannak olyan ablakkezelők, melyek képi világukkal nyűgözik le a felhasználót, mások pedig teljesen elbújnak, egyetlen képpontot sem (!) jelenítenek meg állandó jelleggel a képernyőn, még az ablakok címsorait sem. A változatosság gyönyörködhet, a felhasználók kiválaszthatják az igényeiknek és tudászintjüknek megfelelő ablakkezelőt. E sorok szerzője az fvwm2-t választotta, több napot eltöltött a beállításokkal, de nem bánta meg, mert minden kattintás kézre áll, és minden ablak szinte magától a helyére kerül, a takart ablakok közt nem kell vaktában keresgélni. Ha valakinek nincs ideje hosszas konfigurálásra, az icewm-et érdemes használnia, és ha van egy szabad órája, akkor végigpróbálgatnia az iceconf nyújtotta beállításokat.

A widget-készlet főleg általános célú widget-ek (felirat, nyomógomb, szövegbeviteli mező, scrollbar, menü, toolbar stb.) egységes rendszere, amely nem csak kirajzolni képes a widget-eket, hanem billentyűzet- és egéreseményeket is fogad, lehetőleg magától értetődő és konzisztens módon (pl. az ablak tetejéről legördülő, és az ablak közepén előugró menükben is ugyanazokkal a billentyűkkel lehet navigálni). Windows és MacOS esetén minden alkalmazás ugyanúgy néz ki, és ez a közös widget-készletnek köszönhető. A Windows XP ugyan tartalmaz merész újításokat (melyek főleg a widget-ek méreteit és kinézetét érintik), de az újítások az XP alatt futó összes programra érvényesek. X11 esetén viszont szinte minden program máshogy néz ki, mivel nincs szabványos widget-készlet. Ez a sokrétűség összezavarja a felhasználót, és gátja a hatékony munkának. Hiába használunk pl. KDE-t, ami önmagán belül egységes, ha a PDF-nézegetőnk, az Acrobat Reader Motif widget-készleten alapul, és mind megjelenése, mind kezelése merőben elüt a többi KDE-s programétől. A szerző véleménye szerint a legszebb szabad widget-készlet a GTK, amely ezen túl praktikus és modern is (lásd a következő fejezetben). A rivális widget-készletek összefogása a közeljövőben nem várható.

4. Összehasonlítási szempontok

A grafikus felületek tervezése igen kényes feladat, mert eltérő igényű és tudásszintű felhasználók elvárásainak kell megfelelni, és egy nevelő célzatnak is meg kell jelennie, hogy a felhasználó napról napra okosabban és hatékonyabban legyen képes használni a felületet. Igazodni kell továbbá az eltérő hardverkörülményekhez (kevés memória, kevés szín megjelenítésére képes videókártya, lassú hálózati kapcsolat) is. A képi marketing szempont is jelentős, sokan a egy mintaképernyő (screen shot) alapján szűkítik első körben a szóba jövő grafikus felületek körét, továbbá szép környezetben a munka is könnyebben megy. Sajnos ezen a rostán az fvwm2 fennakad, mert alapkiépítésben fapados és csúnya. A Windows szerencsés, mert többségi részesedése miatt a maga képére alakíthatta a felhasználók izlésvilágát. Ha valami máshogy működik, mint Windows-ban (pl. máshogy, máshova kell kattintani), az megazavarja a felhasználók nagy részét. Kétségtelen, hogy a Windows (különösen a Windows 95)

widget-készlet tekintetében maradandót alkotott. (A másik, széles körben elismert érdeme a TrueType fontok bevezetése.) A Windows-os widgetek szépek, áttekinthetőek (talán a 3D-s benyomásnak köszönhetően), keveset foglalnak a képernyőn, billentyűzetről vezérelhetők, és az alap widget-típusokból majdnem minden összerakható. A GTK sok tekintetben a Windows-os widget-ek előnyös tulajdonságainak nyomdokaiban jár, de a kevés helyfoglalásra a fejlesztők nem ügyelnek: a toolbar-ok és a dialógusdobozok terpeszkednek a képernyőn. A grafikus felület egyéb részeit tekintve minden mai megoldásnak van mit tanulnia a többitől.

4.1. Az ablakkezelő feladatai

Ablak mozgatása. Windows alatt ezt a címsor megragadásával (drag and drop) tehetjük meg. De mi van, ha a címsor épp kilóg a képernyőről? Ekkor az Alt-⟨Space⟩-re előjövő rendszermenüvel kell ügyeskednünk. UNIX-os ablakkezelők gyakori és hasznos megoldása, hogy az ⟨Alt⟩ nyomvatartása mellett az ablakot bárhol megragadhatjuk. Alapvető igény, hogy az ablakok kilóghassanak a képernyőről, így megszabadulhatunk a buta programok menü- és gombsorától, melyek az érdekes dokumentum elől foglalják a helyet.

Ablak átméretezése. Ez egy ritkább művelet, mint a mozgatás, a legtöbb ablakot eredeti méretben vagy maximalizálva szoktunk használni. Épp ezért hasznos lehet az a beállítás, hogy az ablakkeret oldalai mozgatásra, a sarkai pedig átméretezésre valók. A téves átméretezés nehezebben visszaállítható a téves mozgatásnál, ezért is jó, ha csak a sarkakban lehetséges.

Új ablakok okos elhelyezése. Jó elhelyezési stratégia, ha az új ablakot balról jobbra, felülről lefelé próbálja elhelyezni a gép, ha van hely úgy, hogy más ablakokat ne takarjon. Ha nincs hely, akkor például a bal felső sarokból átlósan jelenjenek meg az új ablakok, hogy legalább a címsora mindegyiknek látszódjon. Sokan felhasználó szereti beállítani, hogy minden új ablakot neki, kézzel kelljen leraknia, de ez zavaró lehet, ha egy popup ablak épp akkor jelenne meg, amikor a felhasználó épp mással foglalkozik.

Virtuális asztal (virtual desktop). Az asztal lehet nagyobb a képernyőnél, és az ablakkezelő lehetővé teheti, hogy a képernyőt, mint periszkópot mozgassuk a nagy asztal fölött. Ha mindig épp egy képernyőméretnyit mozdulunk odébb, úgy tűnhet, mintha több asztal lenne, és azok között váltogatunk. Az is megoldható, hogy a szomszédos képernyődarabokról lelógó ablakok ne jelenjenek meg. Fontos megfigyelés, hogy a virtual desktop-ok támogatása nem igényel többletmemóriát, mivel az ablakkezelőnek csak azt kell megszerveznie, hogy asztalváltásnál és -tologatásnál bizonyos ablakok eltűnjenek (minimalizálódjanak), mások pedig megjelenjenek. Virtuális asztalok esetén nincs szükség az ablakok közti körbejárásra (Windows alatt Alt-⟨Tab⟩), mert minden ablaknak vagy ablakcsoportnak dedikált asztala van, és az ablakok nem takarják egymást. A virtuális asztalokat nevük és/vagy számuk különbözteti meg egymástól, így a felhasználó biztosabban megtalálja őket, mintha a tálcán kézzel kéne az ablakokat „vaktában” csukogatnia és nyitogatnia. Munkahelyi lustulás esetén is kapóra jöhetnek a virtuális asztalok: egy asztalon kikapcsolódunk, a másikon pedig nyitvatartunk néhány munkaablakot, hogy gyorsan átválthassunk rá, ha benyit a főnök.

Ablak maximalizálása, minimalizálása (ikonizálása). Ezen műveletek jelentősége a virtuális asztal használatakor csökken, mert minden program a saját virtuális asztalán maximális méretben futhat.

Különböző fókuszmodellek. Windows alatt a fókusz nem változik az egér mozatásának hatására (click-to-focus). A Tweak UI segítségével beállítható, hogy mindig az az ablak kapja a fókuszt, amely fölött az egér van, tehát ne kelljen külön kattintani (focus-follows-mouse). A legtöbb UNIX-os ablakkezelő a felhasználóra bízza választást a kétféle fókuszmodell között. Az utóbbi előnye, hogy egy kattintást megspórolhatunk, és ha a hosszú billentyűzések között csak ritkán nyúlunk az egérhez, akkor nem kell precízen megragadnunk, elég éppen csak odébblökni, hogy egy másik ablak legyen aktív. Továbbá az is megtehető, hogy az elöl lévő ablakot nézzük, de a mögötte levőbe gépelünk. Ennek a megoldásnak a szemére vetik, hogy az egér gyakran a szándékunk nélkül is mozoghat, például ha véletlenül meglökjük. A szerző több, mint 5 év grafikus számítógéphasználat során egyszer sem lökte meg az egeret.

Automatikus ablakemelés. Gyakori választható funkció, hogy az egér alatti ablak kattintás nélkül a többi fölé emelkedjen, ha az egér elidőzik fölötte. Ez veszélyes is lehet, mert pusztán az egér mozgatása tönkreteheti a gondosan és fáradságosan kialakított takarási sorrendet. Nem teljesül továbbá az a hallgatólagos elvárás, hogy ha az egeret kattintás nélkül odébbvisszük, majd vissza, annak ne legyen hosszútávú következménye.

Ablakok mélységbeli mozgatása. Gyakori, hogy kíváncsiak vagyunk, mi van egy adott ablak mögött. Sok ablakkezelőnél az Alt-jobbegérgomb kombinációt az aktuális ablakot legalulra, az Alt-balegérgomb kombináció pedig legfelülre helyezi. Ezek a kombinációk célratörőbbek, mint a Windows Alt-⟨Tab⟩ körbejáró funkciója, nem is beszélve arról, hogy megvalósítható velük az összes böngészőablak egyszeri, konzisztens körbejárása, míg az Alt-⟨Tab⟩ az utolsó kettő között oszcillál, és a többinek még a címsorát sem mutatja meg.

Stílusok. Az egyes ablakok beállításai (például legyen-e címsor, átméretezhető legyen-e, az ablakkezelő mekkora kezdőméretet ajánljon, a keret színe) alkalmazásonként eltérőek lehessenek. Általában a legtöbb alkalmazást ugyanarra a stílusra érdemes állítani, csak néhánynál érdemes kivételt tenni.

Ne pazarolja a helyet. A mindennapi munka során általában a képernyő összes képpontjára szükség van, ezért nem kívánatos, ha az ablakok keretei, és az ablakkezelő által megjelenített egyéb részek elveszik a helyet a fontos tartalomtól. Illusztrációképpen képzeljük el, hogy 640 × 480-as vagy 800 × 600-as felbontásban egy böngészőablakot maximalizálva indítva a képernyő hányadrészét tölti be a weboldal. Már a képernyő fele is jó aránynak számít, emellett a start menüt, a címsort, a menüsort, a gombokat, a különböző haszontalan füleket, a „kedvenceket” és a státuszsort kell bámulnunk. Minden widget csak egy kis részét hasítja ki a képernyőnek, de az adók és járulékok után csak a terület fele marad használható. Az ablakkezelő részéről pazarlást jelent a vastag keret, a nagy címsor, és az óriási gombok. A Windows XP-ben például ezek mind lazábbak a Windows 95-ben megszokott elegáns tömörségnél. A szerzők valószínűleg azzal érvelhettek – tévesen –, hogy a „noha a monitorok felbontása megnövekedett, a felhasználó továbbra sem kíván rajta több információt látni”. A KDE nagy előnye, hogy az ikonok és egyéb dekorációk mérete finom lépésközzel testreszabható.

4.2. A widget-készlet feladatai

Legyen konzisztens. A felhasználók igénylik, hogy a különböző programokban ismétlődő azonos widget-et egyformán nézzenek ki és egyformán reagáljanak az eseményekre. A Windows-nak már a 3.1-es verzió óta része egy jól használható, szabványos widget-készlet, melynek megjelenése kisebb változásokon ment át az évek során, de a bővülésen kívül viselkedése nem változott, és kezeléséhez a legtöbb felhasználó már hozzászokott. Ez az egység nem jellemző az X11-es világra, ahol az Athena, Xaw3D, Motif, Tk, GTK, Qt, Fltk, Gecko, Swing widget-készletek mellett számos gyártóspecifikus és nyílt megoldás is született. Ezek között sem a vizuális összhang, sem a kompatibilitás nincs meg, máshogy viselkedik például a GTK és a Qt szövegbeviteli mező egy ékezetes billentyű megnyomásakor. Arról nem is beszélve, hogy máshogy reagálnak az eseményekre (máshol kell rájuk kattintani, más billentyűkombinációkat várnak), és a sima szövegnél bonyolultabb információ nem másolható a vágólapon. Mindennek a felhasználó issza meg a levét, mert pl. szinte minden alkalmazás szövegablakát másképpen kell görgetnie. Joggal mondhatjuk, hogy az X11 esetén káosz uralkodik. Az GTK és Qt erős versenytársai egymásnak, és a többieknek is van olyan alkalmazási területük, melyen verik a mezőnyt, mindenki tartani igyekszik a pozícióját. A megoldás csak egy új, jól átgondolt, közösen tervezett, és mindenki által előre elfogadott widget-készlet jelenthetné, de a fejlesztők jelenleg nem kívánnak energiát ölni egy ilyen mértékű összefogásba.

Reagáljon a billentyűzetre. Az alkotó, fejlesztő munka nagy részét a fejlesztő billentyűzettel végzi, ezért elvárható, hogy a legtöbb funkció elérhető legyen valamely billentyűkombinációval: minden menüelem és egyéb funkcióhoz legyen hozzárendelve gyorsbillentyű, a felhasználó módosíthassa a hozzárendeléseket, és a dialógusdobozokban is lehessen navigálni (⟨Tab⟩, ⟨Enter⟩, ⟨Esc⟩ és ⟨Space⟩ billentyűk). Ebben persze a grafikus alkalmazás fejlesztőjének is közre kell működnie, például lehetőséget kell biztosítani a gyorsbillentyű-hozzárendelések file-ba mentésére. A gyorsbillentyűk tekintetében a GTK a legpraktikusabb: a menüelemekhez rendelt gyorsbillentyűket a felhasználó menet közben módosíthatja,

úgy, hogy egérrel kiválasztja a menüelemet, majd lenyomja a kívánt kombinációt. A változtatások sajnos elvesznek, ha a fejlesztő nem működik közre. Az Fltk mindig figyel arra, hogy a megfelelő ablak legyen felül, és az kapja meg a fókuszt, és hogy az ablakon belül is legyen aktív widget, már az ablak megjelenésekor. Ehhez, ha szükséges, az egérkurzort is odébbmozgatja. Továbbá minden képességét latba veti, hogy billentyűesemény ne vesszen el: gondosan több ablaknak és widget-nek is felkínálja, hátha tud vele kezdeni valamit. Sajnos a legtöbb widget-készletnél nem ez a helyzet: például a Motifon alapuló Netscape Navigator elindulása után nem írhatjuk be rögtön a meglátogató weblap címét, mert előbb a *Location:* beviteli mezőre kell kattintanunk. A sok hasonló kényelmetlenség végül azt eredményezi, hogy a felhasználónak az egér és a billentyűzet között kell kapkodnia.

Legyen tetszetős. A vonalvastagságok, az arányok, a betűtípusok és a betűméret és a néhány alapszín egymáshoz illő megválasztása nem csak esztétikai szempontból fontos, hanem olyan körülmény, amely a munka hatékonyságát is befolyásolja. Nincs szükség színátmenetekre vagy háttérképekre, purintán építőelemekből is létrehozható gyönyörű widget-készlet. A Windows 95 és a GTK szépnek mondható, a Qt inkább modernnek és csillogó-villogónak. A GTK és Qt widget-ekre kölönböző bőrök (skin) húzhatók, a felhasználó izlésvilágának megfelelő kinézetet választhat, például a Qt-nek van Windows 95-szerű és Motif-szerű skin-je is. A skin nem nevezhető emulációnak, mert csak a kinézetet emulálja, a viselkedést (pl. a gyorsbillentyűket) nem. Fontos lehet, hogy a skin-ek ne befolyásolják a widget méretét, mert ellenkező esetben a programozó nem lehet biztos abban, hogy egy dialógusdoboz widgetei mind megfelelően látszódnak-e, nem takarnak bele egymásba. Ez a méretkorlátozás ún. geometry manager-ekkel (más néven: layout manager) megoldható, melyek intelligensen, relatívan, a logikai kapcsolatokat figyelembe véve helyezik el a widget-eket, és az ablak méretét a widget-ek és a köztük levő kihagyások összhelyigényéből származtatják. Így a takarás és az ablakból kilógás megszűntethető, de a fejlesztő már nem lehet biztos abban, hogy az ablakai mindehol *szépek* lesznek. X11 környezetben már az eltérő fontbeállítások is otromba ablakokat eredményeznek.

Tab-sorrend. Fontos, hogy az egy ablakban levő widget-ek billentyűzetről is bejárhatóak legyenek. Az ⟨Enter⟩ általában az *OK* gombra, az ⟨Esc⟩ pedig a *Cancel* gombra klikkeléssel egyenértékű, a listákban nyílbillentyűkkel lehet haladni, és szóközzel lehet kiválasztani. A widget-ek között egy körkörös sorrend van definiálva, melyet a ⟨Tab⟩ illetve Shift-⟨Tab⟩ lenyomásával járhatunk végig előre és hátra. Minden ablak megjelenésekor kell lennie egy induló widget-nek. A hasznos popup ablakoknak (például rákérdezés kilépés előtt) úgy kell megjelenniük, hogy rögtön magukhoz ragadják a fókuszt, hogy még ekkor se kelljen a felhasználónak az egérhez nyúlnia. Általában elmondható, hogy az Fltk sok figyelmet szentel a billentyűzetnek, mögötte a Windows és a GTK következnek, majd a Qt és a többiek. Ár ellen evezésnek tűnhet a mai egerészős világban az okos billentyűzethasználat mellett kardoskodni, de aki alkotó munkát vagy adatbevitelt végez, tehát nem csak fogyasztói, befogadói szinten használja a számítógépet, annak a billentyűzet az egérnél gyorsabb és fontosabb munkaeszköz. Sajnos a mai widget-készletek nem sarkallják arra a felhasználókat, hogy a navigációt billentyűzetről végezzék. Szinte mindig valami miatt oda kell nyúlni az egérhez, pl. az rossz helyen feldobódó eszköztárakat kell únos-untalan odébbmozgatni.

Betűtípus-választás. Igen szerencsétlen probléma, hogy hiába van egy betűtípus feltelepítve a gépünkre, ha speciális neve, vagy peraméterei miatt nem tudjuk kiválasztani. Például ha a betűvastagság félkövér és normál között van, akkor a hagyományos betűtípusválasztó dialógusdobozokban ezt fontot nem tudjuk megadni. Hasonló probléma van mind GTK, mind Qt esetén: a hagyományos 6×13-as betűtípust nem lehet kiválasztani. Ez felettébb kellemetlen, mert ez a betűméret lehetővé teszi 1024×768-as felbontásban 4 db XTerm megjelenítését a képernyőn úgy, hogy azok ne takarják egymást, és a betűk még olvasható méretűek legyenek. A GTK-ban és Qt-ben beállítható többi betűméret vagy túl kicsi, vagy csak egyetlen XTerm fér el, de ekkor a munkahatékonyság csökken, mert nem fejleszthetünk az egyik terminálablakban, miközben a másikban a szükséges dokumentációt olvassuk. Sajnos a grafikus felületek fejlesztői sokszor vezetnek be ilyen önkényes korlátozásokat (pl. nem lehet tetszőleges betűtípust beállítani), nem is gondolva arra, hogy ezzel sokak mindennapi munkáját nehezítik. Szerencsére az XTerm fontbeállításait nem egy dialógusdobozban kell elvégezni, elég, ha `xterm -font -misc-fixed-medium-r-semicondensed-13-120-75-75-c-60-iso8859-*`-del indítjuk (ebből a a `semicondensed`-et nem ismerik a fontválasztó ablakok).

Animációs effektek. Egy menü vagy a tree widget egy ágának lenyílása, egy ablak minimalizálása vagy a szöveggörgetés is megvalósítható több fázisban, animációval. Ezek a jópofa effektek fokozzák a vizuális élményt, de a hatékony munkát gátolják: például sokkal lassabb is idegesítőbb átfutni a start menüt, ha minden almenü lenyílására egy tizedmásodpercet várni kell. Engedélyezésüket a felhasználóra kell bízni. A Windows 98 kezdveli az ilyen effekteket, és csak rosszul dokumentált registry-buherálással lehet ezeket lebeszélni erről. Az X11-es widget-készletekben általában nincs ilyen effekt, vagy menüből állítható. Az animáció a hálózati teljesítményt is rontja. Képzeljük el például a magyar felsőoktatási intézményekben használatos Neptun rendszer további lassulását, ha minden ablak becsukásakor végig kéne néznünk egy több másodpercre nyúló értelmetlen animációt.

Információ elrejtés és megmutatás. Ha egy ablakban túl sok információ vagy beállítási lehetőség szerepel, akkor az ablak mérete túl nagy lesz, és a felhasználó nem igazodik el rajta. Túl kevés információnál viszont a felhasználó állandóan a füleket és a menüpontokat lapozgatja az őt érdeklő funkciót keresve. Gyakori kompromisszumos megoldás egy tree widget alkalmazása, melyben könnyű navigálni, és a felhasználó akár az egészet kinyitogathatja, egészen a levelekig. Ha egy ablak nem csak szabványos widget-eket jelenít meg, hanem például egy képet, ábrát, vagy szöveges dokumentumot, akkor azt az ablakot átméretezhetővé és nagyíthatóvá (zoom) kell tervezni, és nem szabad benne semmilyen más widget-et elhelyezni. A felhasználó, ha úgy látja jónak, dock-olhasson rajta eszköztárakat. Az is kifejezetten rossz gyakorlat, hogy a formázó ablakokat a dokumentum fölé, a képernyő szélére helyezik. Ekkor a felhasználónak állandóan scroll-oznia kell, hogy lássa a dokumentumot. Ehelyett ajánlott a formázó ablakokat eszköztárrá konvertálni, és az eszköztár egyes gombjai menüket hívjanak elő. A GIMP képszerkesztő program egy másik jó megoldást követ: minden fukció elérhető a jobbkattintásra előugró context menu-ből, továbbá testreszabható billentyűkombinációkkal. A képernyő mérete relatíve kicsi a szerkesztendő dokumentumokhoz és a képernyő felbontásához képest. A legrosszabb, amit egy alkalmazás tehet, hogy saját, ritkán használt és fölösleges ablakaival pakolja tele a képernyő alkotásra fenntartott részét. Nem ajánlott az Acrobat Reader viselkedése sem, amely a dokumentum margóját is megjeleníti, és így a felhasználó fehér felületek által közrőfogott apró, olvashatatlan betűket kénytelen bámulni, vagy pedig únos-untalan görgetnie kell.

4.3. Egyéb, járulékos feladatok

Start menü. Ebben a menüből indíthatók a gépre telepített, a felhasználó által hasznosnak ítélt programok. Fontos tulajdonsága, hogy minden program parancsikonja telepítésekor belekerül a start menübe, de a felhasználó a parancsikonokat később tetszés szerint módosíthatja, de akár újakat is felvehet. UNIX és X11 alatt általában a start menü nem bővül automatikusan, így nem tükrözi, hogy a gépre milyen programok vannak telepítve. A kézi karbantartás egyedül arra teszi alkalmassá, hogy a leggyakoribb programok indításához ne kelljen a billentyűzethez nyúlni. A start menü gombjának nem kell folyamatosan látszódnia, sokkal praktikusabb, ha az asztal üres részére kattintva előugrik a start menü, vagy ha egy billentyűkombinációval (Windows alatt Ctrl-⟨Esc⟩) hívjuk elő.

Kézi programindítás. A programokat a szöveges felhasználói felületeknél a program (parancs)nevének beírásával indíthattuk. Ez az indítás elérhető a Windows start menüjének *Futtatás* pontjában, de ez a szöveges shell-ek számos előnyét nem biztosítja (például keresési útvonal állítása, aktuális könyvtár megjegyzése, keresés a history-ban, filenevek kiegészítése, visszalapozás biztosítása hosszú kimenet esetén stb.) Természetesen szöveges parancsokat is indíthatunk így, ezek egy új konzolablakban (DOS ablak) fognak futni. Sajnos, ha egy program túl korán leáll, az ablak eltűnik, mielőtt végigolvassuk a program kimenetét, ezért a start menüből a `command.com` vagy (Windows NT-től fölfelé) a `cmd.exe`-t érdemes futtatni, és a többi programot innen indítani. A `cmd.exe` számos jó ötletet átvett a UNIX shell-ektől, például a history-t és a filenév-kiegészítést, igaz, sokkal gyengébb szinten. Az ékezetet vagy szóközt tartalmazó filenevek kezelése még ekkor is nehézkes, és a parancsok gépelése is hosszúra nyúlhat, mert a Windows nem biztosít egy olyan keresési útvonalat (`$PATH`), melyen minden program rajta van. UNIX esetén az egyetlen szöveges program, amit a start menüből érdemes futtatni, az az `xterm`, a többi lehet, hogy el sem indul, vagy a háttérben lefut, de a szöveges kimentete nem jelenik meg. Az XTerm-ben megkapjuk a beállított shell-ünket, ahonnan a megszokott kényelemből hívhatjuk meg a többi programot. A shell-ből grafikus programok is indíthatóak, ezeket érdemes a parancssor

végére rakott `&` jellel vagy később, Ctrl-⟨Z⟩ lenyomásával a háttérbe rakni, majd `disown` paranccsal a shell felügyeletén kívül helyezni (hogy az XTerm bezárásakor ne csukódjon be aa többi ablak). Annak, aki sokat dolgozik XTerm-ben, nincs szüksége a start menüre.

Ikonok az asztalon. A start menün és a program nevének begépelésén túl még az asztalon található ikonok is használhatók programindításra. Ennek csak korlátozott haszna van, mert az asztalt hamar elborítják a nagyobbnál nagyobb ablakok, és az ikonok előrehozása romba döntheti az ablakok precízen kialakított mélységsorrendjét. Épp ezért érdemes inkább a start menüben egy külön listát létrehozni a gyakran szükséges programok számára, ezzel az asztal ikonjai kiválthatók.

Doumentumok megnyitása, asszociációk. Windows-ban a file-ok típusát kiterjesztésük határozza meg: nevük mellett a kiterjesztésüknek megfelelő ikon jelenik meg, és a kiterjesztésük dönti el, hogy melyik program nyissa meg őket, ha rájuk kattintunk. Ez utóbbit asszociációnak nevezzük. Többfajta (pl. megnézés, szerkesztés) asszociáció rendelhető egy kiterjesztéshez, és minden fajtában van egy elsődleges, a többi csak a jobbkattintásra előugró context menu-ből érhető el. UNIX alatt az asszociáció egy külső információn, a MIME-típuson (MIME type) alapul. Például a `text/html` típusú file-t egy webböngészőben, a `video/mpeg` típusút pedig egy médialejátszóban kell megnyitni. A felhasználó a `~/.mailcap` file-ban módosíthatja az asszociációkat, és a `~/.mime.types`-ban pedig azt adhatja meg, hogy a külső MIME-típus hiánya esetén a file kiterjesztéséből hogyan kell kikövetkeztetni a típusát. A Midnight Commander ezen alapuló, de ennél finomabb asszociációs módszert alkalmaz, amely a *Command* menü *Extension file edit* pontjában írható át. Örvendetes, hogy egy file megnyitásához nem kell ismerni a szükséges alkalmazás nevét. Ez azonban veszélyeket is felvet, például ha egy Java forráskódhoz az egyik nehézsúlyú Java fejlesztőeszköz van rendelve, akkor egy óvatlan duplakattintás hatására egy percre használhatatlan lesz a gép, és a lemez zakatalosát leszünk kénytelenek hallgatni.

Az ablakok távoli megjelenítésének biztosítása. Az X11 architektúrája ezt alapból lehetővé teszi, Windows alatt a VNC segíthet, de ettől még tud két felhasználó egyszerre bejelentkezni. Az igen drága Windows Terminal Server a párhuzamos bejelentkezéseket is lehetővé teszi.

File-kezelés. Elvárható, hogy a file-ok böngészése, a keresés, a másolás, a mozgatás és a törlés egyszerű legyen, és a felhasználó olyan nézetben lássa file-jainak nevét és egyéb paramétereit (pl. méret, utolsó módosítás dátuma), hogy hamar kiválaszthassa a következő műveletben résztvevő file-okat. A másolás és a mozgatás két helyhez is kötött (honnan és hová), ezért értelemszerűen legalább két ablakra van szükség ezen műveletek vizuális elvégzéséhez. A vizuális file-műveletek úttörője és nagy öregje a Norton Commander, amely szöveges módban, két egymás melletti két panellel dolgozott. Mindkét panel egy-egy könyvtár tartalmát mutatta (nem teljes mélységben), tehát olyan volt, mint a Windows Intéző a bal oldali fa nélkül. Voltak formátumfüggő műveletek is, például megnézés és szerkesztés, ezekhez lehetőség volt külső program meghívására is. A felhasználó saját menüt definiálhatott, a menüből hívott programok pedig paraméterül megkaphatták a kijelölt file-ok neveit. Ily módon a file-okon *tetszőleges* művelet elvégezhető volt. A DOS-os Norton Commander-nek egy orosz szerző elkészítette a 35 kB-os (!) változatát, ami sok tekintetben többet tudott az eredetinél. Linux alatt a (szöveges módú) Midnight Commander a legnépszerűbb klón, a UNIX-filozófiába illeszkedő testreszabási és egyéb lehetőségekkel. Az X11-en alapuló desktop environment-ek számos gyenge file-kezelőt kitermeltek, melyek nem vetekedhetnek a Norton Commander-klónok jól átgondolt, hatékony egyszerűségével. Az új file-kezelők a Windows Intézőjét próbálják majmolni, de mögötte kullognak, és jó megjelenésük ellenére csak lassan és körülményesen használhatók. A UNIX hacker-ek szeretik parancssorból karbantartani a file-jaikat, mert ekkor érzik őket biztonságban. Ez a módszer nem olyan fapados, mint DOS alatt, mert a shell mind a filenév-kiegészítéssel, mind a joker karakterekkel segíti a file-ok kiválasztását.

Windows alatt az eredeti színvilág és billentyűkombinációk kedvelőinek az FAR Manager, a grafikus felületet szeretőknek pedig az Windows Commander, más néven Total Commander ajánlott. A Windows részeként alapból érkező intéző komoly munkára nem ajánlott, mivel bizonyos könyvtárakat (pl. a Windows alapkönyvtárát) alapból nem mutat meg, a nagybetűs és kisbetűs fileneveket összevissza keveri, elrejti a kiterjesztéseket, új verziói pazarolják a képernyőterületet, a file asszociációkat bután kezeli, és a háttérben végzett feladatok (pl. könyvtárak méretének összeszámolása, az ikonok betöltése) rontják a válaszidejét. A felhasználónak a Windows intéző esetében gyakran van olyan érzése, hogy az elrejti, elfedi, eltitkolja az információt, ahelyett, hogy a lehető legprecízebben megmutatná.

Tálca. Az ablakkal rendelkező, futó folyamatok (process) listáját Windows-ban a tálca szűk, egysoros listáján követhetjük nyomon, és a kívánt ablakot egy kattintással előrehozhatjuk. A tálca azonban a mindennapi használat során betelik, és csak annyit látunk, hogy van 5 db I-vel kezdődő böngészőablakunk, és 8 db M-mel kezdődő szövegszerkesztőnk. Ezután igen nehéz a első kattintásra eltalálni azt az ablakot, melyre át akarunk váltani. A Windows XP csoportosítással javított a helyzeten, de még mindig nehézkes a navigálás. A tálcát icewm esetén kiegészíti, fvwm2 esetén helyettesíti egy többsoros ablaklista, amelyben egymás alatt láthatjuk a nyitott és minimalizált ablakok címsorait, és választhatunk is belőlük. Az ilyen típusú ablaklistát gyakran fölöslegessé teszik a virtuális asztalok, mert a gyakorlott felhasználó az összetartozó ablakokat azonos asztalon helyezi el, vagy takarás nélkül, vagy – nagy ablakok esetén – takarással, és ez utóbbi esetben körbepörgetéssel (Alt és jobb egérgomb) találja meg a neki kellőt, az átugrott ablakoknak nem csak a címsorát, hanem a tartalmát is látva.

Átlátszó ablakrészek, nem téglalap alakú ablakok. Ezek első látásra jól néznek ki (pl. mobiltelefon-alakú WAP-böngésző), de nincs gyakorlati hasznuk, és az is bosszantó, hogy az ilyen ablakok többet foglalnak, mintha hagyományos keretűek lennének, téglalap alakkal. Az ilyen ablakok megjelenítése és a képfrissítés érezhetően lassabb, hálózaton áthozott grafikus felületnél katasztrofális lehet. Örvendetes lenne, ha a programozók a felhasználóra bíznák a döntést, hogy szeretne-e ilyen ablakot látni az általa futtatott programokban.

Áttetsző ablakrészek. Az is igen modern, ha egy terminálablak háttere félig áttetsző. Ennek a számításigényét akkor tapasztaljuk meg, ha az ilyen ablakot odébbmozgatunk. Több tizedmásodpercet kell válni, mire a háttér visszaköveti a mozgást. Az ilyen vizuális effektek igen megnehezítik a gép interaktív használatát, ha az egyébként le van terhelve. Gondoljunk arra, hogy nyitnánk egy terminálablakot, hogy leállítsunk egy elszabadult, az erőforrásokat elfogyasztó folyamatot, de a terminálablak megnyitásához is extra erőforrásra van szükség, tehát kénytelenek vagyunk percekig (!) hallgatni, amint a gépünk fölöslegesen swap-pel, és eddig semmi másra nem tudjuk használni.

Konfigurálhatóság. A felhasználók egyéni elvárásaik vannak a grafikus felület viselkedését illetően. A felületnek részletes, könnyen elérhető dokumentációt kell biztosítania a módosítható részekről, és egy konfigurációs ablakban vagy file-ban lehetőséget kell biztosítania a beállításra. Windows esetén ez az ablak vezérlőpult, melyben nem csak a grafikus felület, hanem az egész rendszer viselkedését befolyásolhatjuk. Ilyen nagymértékű integráció UNIX és X11 esetén nem figyelhető még, a KDE és a Gnome vezérlőpultjai csak a saját hatáskörükbe tartozó, tehát a desktop environment-et érintő beállításokat végzik el megbízhatóan, s noha egyéb lehetőségeket felkínálnak, ezek a külső beállítások gyakran hatástalanok. Ez törvényszerűen van így, hiszen a desktop environment-ek és a UNIX-szerű operációs rendszerek heterogén környezetben, párhuzamosan fejlődnek. UNIX alatt a felhasználónak általában a man page-eket kell olvasnia, újabb programok esetén az interneten elérhető, legfrissebb dokumentációt, és ez alapján kell a megfelelő névvel és tartalommal rendelkező konfigurációs file-t begépelnie. E fáradságos munka utáni elégtétel, hogy a felhasználó pontosan tudja, mit akar; újratelepítéskor vagy költözéskor elég csak másolni a konfigurációs file-okat, továbbá sok tekintetben nagyobb szabadságra és automatizálásra van lehetősége. A jó dokumentáció eleje gyorsan olvasható, a vége kimerítő és részletes; működő és nem működő példákat tartalmaz (utóbbiak esetén indoklással és javítással), illeszkedik a szoftver aktuális verziójához. A dokumentáció mindig az okosabb felhasználókat (power user) kell, hogy megcélozza, és új információt kell tartalmaznia (pl. az „IP-cím” mező súgója ne pusztán annyi legyen, hogy „itt adja meg az IP-címet”).

4.4. A szerver feladatai

Válaszidő nagy ablakok esetén. Ha egy ablak háttérképet tartalmaz, akkor területével arányos memóriát igényel. Bizonyos programok, például a Ghostscript néha óriási, tízezres méretű ablakokat akarnak nyitni. Ekkor a gép elkezd swap-pelni, és meg kell várni azt az 1–2 percet, amíg a swap is betelik, és a Ghostscript kilép. Sajnos a mai X szerverek nem figyelnek a korlátozott erőforráshasználatra, és a gyors válaszidőre, így már az elszabadult programokat kilövő parancs beírása sem lehetséges a rendszer terheltsége miatt. Hasonló a helyzet Windows NT esetén, ahol ugyan a Ctrl-Alt-⟨Del⟩ lenyomása után azonnal kiválaszthatjuk a Task Manager-t, de az néha csak percek múlva jelenik meg, így a gép fölötti uralmat csak időkiesés árán szerezhetjük vissza. X11 alatt megadatott a gyors kilépés lehetősége

(Ctrl-Alt-⟨BackSpace⟩), ami néhány másodperc alatt az összes reagáló grafikus programot bezárja, és ez gyakran azonnal megszűnteti a terhelést. Sajnos ekkor a fontos, de mentetlen adatok elvesznek.

Az egérkurzor árnyéka. Teljesen fölösleges, a hatékonyságot rontó vizuális effekt. Lehetőséget kell biztosítani arra, hogy a felhasználó kikapcsolja. Helyette hasznosabb lenne, ha az egérkurzor villogni hezdene az egér összevissza rángatásának hatására, így gyorsabban meg lehetne találni a képernyőn.

Unicode-os betűk. Manapság nem csak angol nyelvterületen használnak számítógépet, ezért a szöveges widget-eknek ismeriük kell az összes szóba jövő betűt, ezért érdemes a Unicode-hoz igazodniuk. A Windows a Windows 95 óta támogatja a Unicode-ot, nem csak a widget-ekben, hanem pl. az Office alkalmazásokban is. Az X11, a GTK és Qt-nak csak a legfrissebb verziói Unicode-osak, de a heterogén klienskörnyezet miatt néha nem lehet úgy beállítani a billentyűzetet, hogy a magyar ékezetes betűket minden program elfogadja: kézzel kell kapcsolgatni a logikailag ekvivalens, de technikailag különböző kódokat generáló kiosztások között. Nem elég azonban a helyes kódolás megválasztása, szükség van olyan betűtípusokra (font), melyek kellően sok karaktert tartalmaznak. Windows esetén a szövegszerkesztésben használt *Times New Roman* és *Arial* ilyenek (több, mint 400 karakterrel), míg a szöveges widget-eknél kis méretben használt két betűtípus, az *MS Sans Serif* és a *System* technikai okokból csak 256 karaktert tartalmazhat, fix kódolásban. A Windows XP-ben megjelent *Microsoft Sans Serif* igyekszik kiváltani ezt a kettőt (szintén 400 fölötti karakterszámmal), de kevésbé olvasható, mint elődjei, és itt a kis méret miatt minden képpont számít. Érdemes még a *Tahoma*-t kipróbálni kis méretben.

X11 esetén a Unicode viszonylag új dolog, nagy hiány van sok karaktert tartalmazó betűtípusokból, de még nagyobb a hiány a Unicode-ra felkészült kliensekből. Hosszú ideje egy átmeneti megoldás terjedt el: a Unicode-ra 256 elemű ablakokat képeznek, ezek a kódlapok, és minden nyelvhez olyan kódlapot állítanak be az összes alkalmazás számára, melyben az adott nyelv összes betűjét tartalmazza. A magyarhoz tartozó kódlap az ISO-8859-2 (Latin-2, közép-európai). Ezután vagy szétdarabolják a betűtípust a kódlapoknak megfelelően (sok lesz az ismétlődő betű), vagy a szerver futási időben válogatja ki a megfelelő karaktereket a nagy file-ból. TrueType és Type 1 fontok esetén ez a futásidejű válogatást az X11 elvégzi, de a képernyős használatra tervezett pixeles fontok esetén nem.

Kilensek közti kommunikáció. Ide tartozik a vágólap, a külső ablakok méretének lekérdezése, és hálózati grafikus felület esetén kommunikációs csatorna biztosítasa (pl. postafiók (mailbox) vagy folyam (stream) alapú). Fontos különbség az X11 és a Windows vágólapja között, hogy X11 esetén elég a szöveget egérrel kijelölni, míg Windows-ban egy *Copy* művelet is szükséges, amely a kijelölt szöveg másolatát a vágólapra helyezi. GTK-ban e két kijelölési forma egyidejűleg létezik, a *Copy* csak a billentyűzetről történt kijelölés után szükséges.

Több monitor egyidejű kezelése. Ha valakinek több monitora van, elképzelhető, hogy át szeretné húzni az egeret az egyikről a másikra, vagy egy ablakot (amelyben lehet akár hardvergyorsított 3D képszintézis is) a két monitor határvonalára akar mozgatni. A két monitor jól jöhet nagyobb ábrák grafikai tervezésénél vagy nagyablakú programok debug-olásánál. A Windows 2000 és újabb operációs rendszerek élen járnak a több monitor kezelésében.

A billentyűzetkiosztás átdefiniálása. Windows alatt ez egyáltalán nem lehetséges, csak az előre elkészített kiosztásokból válogathatunk. (Vannak külső programok, például a magyar fejlesztésű multikey, és az ugyancsak magyar Keyboard Remapper, de ezek nem mentesek a hibáktól.) X11 alatt az `xmodmap` és az újabb `xkbcomp` parancsokkal tetszőleges kiosztás beállítható, de a saját kiosztás elkészítése fáradságos, összetett technikai tudást igénylő feladat (viszont elég egyszer és mindörökre elvégezni). Programozók számára előnyös lehet a szabványos magyar kiosztás helyett egy olyan angol kiosztás, melyen az ⟨AltGr⟩ (jobb ⟨Alt⟩) billentyű lenyomása közben ékezetes betűket képelhetünk a szokott helyen (például a pontosvessző helyén `é` betű). Így a programok forráskódját angol kiosztásban, de a kommenteket és a dokumentációt hirtelen magyarra váltva gépelhetjük. A Windows alatti kiosztásváltás (Shift-⟨Alt⟩ vagy Shift-⟨Ctrl⟩) érzékeny mind az időre, mind a két billentyű sorrendjére, ezért gyakran nem vált át. Amikor egy soron belül többször is váltani kéne, igen idegesítő tud lenni, ha értékes másodperceket kell tölteni egy talán sikerülő átváltással.

5. Esettanulmány: Knoppix 3.2

A Knoppix egy új, Debian-alapú (jelenleg Debian Sid), szabad (GPL), egyfelhasználós (de természetesen multitask-ot és többszörös bejelentkezést lehetővé tevő), CD-ről induló, telepítést nem igénylő, grafikus (KDE 3.1.2) Linux-disztribúció, melyben irodai, webes és fejlesztői programok is helyet kaptak. A CD image letölthető a disztribúció honlapjáról (`http://www.knopper.net/knoppix/index-en.html`). Célközönsége a számítástechnikai alapismerettel rendelkező, a Linux-ot megismerni igyekvő, de a Linux-os telepítések nehézségtől megriadó végfelhasználó. A Knoppix-ot a CD behelyezése után 2 perccel ugyanolyan egyszerűen lehet használni, mint egy előre telepített Windows rendszert, és eközben egyetlen byte helyet nem foglal a merevlemezen.

A Knoppix a KDE legmodernebb verzióját tartalmazza, és a KDE-n kívül is számos egyéb program segíti az egyszerű felhasználó munkáját. Épp ezért kettős elvásunk van a rendszertől: a grafikus felület használhatósága érje el a Windows szintjét, de legyen több beállítási lehetőség, legyen stabilabb, és a hagyományos Linux-os alkalmazások fussanak rajta. A tesztelés során az alábbi előnyös (+), semleges (∗) és hátrányos (−) tulajdonságok fogalmazódtak meg a szerzőben:

∗ A CD-n nem található meg a Gnome desktop environment. Az ablakkezelők közül icewm, twm, larswm, xfce, fluxbox és KDE vannak fent. A széles választékból hiányzik a hacker-barát fvwm2.

∗ Kezdő felhasználóknak a KDE javasolt, talán épp ezért a KDE a Knoppix alapértelmezett desktop environment-je. A KDE sok felhasználói és rendszerbeállítást is magára vállal, a legtöbb funkciót elérhetjük a start menüből és az onnan nyíló *Settings/ Control Center*-ből.

\+ A Control Center-ben sokféle fókuszmodell és automatikus ablakelőrehozás (auto raise) is beállítható.

− A KDE-nek nincs olyan menüpontja, amiből ablakkezelőt változtathatnánk, bár ez kezdőknek nem is javasolt. Ha kísérletezni szeretnénk, marad a következő, nem magától értetődő módszer:

```
Ctrl-Alt-<F3> su - knoppix -c 'startx -e icewm -- :3' <Enter>
```

icewm helyett twm-et, larswm-et, xfce-t és fluxbox-ot is indíthatunk a fenti parancs átírásával. Egy új KDE-t pedig a `-e ...` kihegyásával indíthatunk. A régi és az új ablakkezelő között a Ctrl-Alt-⟨F...⟩ billentyűkombinációkkal lehet váltani. Megnyomásuk után várjunk legalább 2 másodpercet, és csak ekkor próbálkozzunk újra.

− A KDE igen lassan indul el, és legalább 256 Mb RAM szükséges a tűrhető sebességhez.

− A root jelszó alapból üres (!), és a Knoppix boot-oláskor felkapcsolódik az internetre (állandó Ethernet kapcsolat esetén, DHCP-vel), ezzel kiszolgáltatva a gépet támadásoknak. Ha a támadó be tud jutni a gépre, egyúttal rendszergazda jogosultságokat is szerzett.

\+ Alapból nem indul SSH szerver, így indítható: *Start menü/KNOPPIX/Services/ Start SSH...*. Ekkor már kötelező beállítani egy nem üres jelszót a „knoppix” felhasználónak. A root jelszó maradhat üresen, mert úgy van beállítva az SSH szerver, hogy root-ot nem enged be.

− A *Start menü/Run Command*-ból szöveges programok (például ls, mc, fdisk) nem indíthatók. Ha ilyet indítunk, akkor a parancs ugyan a háttérben lefut, de semmilyen hibaüzenetet nem kapunk. A kezdő felhasználó nem tudhatja, hogy a (Start menü/Run Command)-ba először `xterm`-et kell írni, majd a szöveges programokat abból indítani.

\+ A start menü (KDE Panel) eltüntethető, mérete testreszabható: jobbkattintás a jobb alsó sarokban, *Configure Panel*, és szinte mindent beállíthatunk.

\+ A KDE alapból négy virtuális asztalt kínál fel.

∗ A virtuális asztalok között a kissé szokatlan Ctrl-⟨Tab⟩ kombinációval válthatunk, de ez átállítható: *KDE Control Center/Regional/Keyboard Shorcuts/Actions/System*-en belül a *Desktop Switching* és a *Navigation* listát érdemes módosítani.

∗ Az egérkurzor alapból áttetsző. Ezt a *KDE Control Center Peripherials/Mouse* beállítások között ki lehet kapcsolni (*White Cursor*), de ehhez újra kéne indítani a KDE-t, amit nem lehet. Az egyéb átlátszóságok menet közben kikapcsolhatók.

− Ha túl gyorsan próbálunk átméretezni egy ablakot, akkor helyette átvált egy másik ablakra.

− A jó öreg, 6×13-as fontot egyetlen KDE menüben sem lehet beállítani, tehát például nem fér ki 4 db Konsole az 1024×768-as képernyőre, kényelmes betűméretben. Szerencsére az XTerm-nek (de az Rxvt-nek nem) ez az alap betűtípusa.

– Az Euro karakter hiányzik a fontokból.

∗ Normál XTerm-et *Start menü/Utilities/XShells/XTerm*-mel indíthatunk.

– Továbbra sem lehet ikonokat másolni az asztal és a start menü között egérrel *mindkét* irányba.

– A KDE néhány beállításnál kéri, hogy indítsuk újra, de ez nem tehető meg egy kattintással.

– Ha véletlenül a Start menü/Logout "Knoppix" -ra kattintunk, reboot-olódik a gép, és minden módosítás elveszett. Az elvárt viselkedés az lenne, hogy újraindul a KDE.

– A Knoppix-os LaTeX nem képes magyar szöveget elválasztani, és mivel nincs írási jogunk a CD-re, a magyar elválasztást a root-ként futtatott `texconfig` parancs sem képes beállítani. Ehhez számos file-t felül kéne írnunk a CD-n. A Knoppix szerzői előrukkolhattak volna egy olyan megoldással, hogy a CD tartalmát a memóriában lehessen módosítani; az nem nagy baj, ha a módosítások újraindításkor elvésznek.

– A KDE-t csak így tudta e sorok írója újraindítani:

```
Ctrl-Alt-<F2> su - knoppix -c 'startx -- :2' <Enter>
```

Ekkor persze a régi (Ctrl-Alt-⟨F5⟩) és az új (Ctrl-Alt-⟨F6⟩) is tovább fut. Az újból Ctrl-Alt-⟨BackSpace⟩-szel (vagy *Start menü/Logout "Knoppix"*) léphetünk ki, de ha ugyanezt a régivel tesszük, akkor újraindul a gép, és minden módosítás elveszett.

– Lehetséges a home könyvtárunk tartalmát újraindítások között megőrizni (*Start menü/KNOPPIX /Configure/Create persistent...*), de a „varázsló" rögtön a partíció nevét kérdezni, amit egy kezdő felhasználó nem tud megadni. Ehhez először a (*Start menü/KNOPPIX/Root Shell*)-ben az `fdisk -l` parancsot kell megadni, és innen választani egy partíciót (pl. `/dev/hda1`). Nem mindegy, melyiket választjuk, és ha nem járunk el körültekintően, akkor a gépen tárolt összes adat elveszhet. Ha ezt nem akarjuk, akkor arra a kérdésre, hogy a Knoppix formázza-e meg a partíciót, mindenképpen nemmel válaszoljunk. Legközelebb indításkor adjuk meg a `home=scan` boot opciót.

– Lehetséges a konfigurációs beállításokat is lemezre menteni. Ezt minden kikapcsolás előtt meg kell tenni: (*Start menü/KNOPPIX/Save KNOPPIX Configuration*). Ugyanazt a partíciót adjuk meg, mint a home könyvtárak esetén. Indításkor ne feledkezzünk meg a `myconf=scan` boot opcióról.

– Nincs felrakva talk kliens és szerver, más UNIX-ok felhasználóival nem lehet kommunikálni.

+ Az Acrobat Reader 4.0-s változata van telepítve, ami még nem olyan szörnyen lassú, mint az 5-ös.

– Nem lehet magyar ékezetes betűket gépelni, hiába állítjuk be a magyar kiosztást, az ő és ű betű helyett csak kérdőjelet kapunk. A megoldás (az egyszeri felhasználónak erre is magától rá kéne jönnie): `export LC_CTYPE=hu_HU xterm`-et kell indítani, és itt már nem lesznek kérdőjelek.

– A billentyűzetbeállításoknál nem adható meg a kiosztások között váltó gyorsbillentyű, állandóan az egérhez kell nyúlnunk, ha váltani akarunk.

– Az alap bash shell nem jelzi ki az aktuális könyvtár teljes nevét, ezzel bizonytalanságban tartva a felhasználót. Sok hiba forrása lehet.

– A Konsole nem követi azt a régi X11-es és XTerm-es hagyományt, hogy a vágólapra másoláshoz elég az egérrel kijelölni a szövegrészt: még egy *Copy* művelet is szükséges a menüből.

– Végig kell kattintgatni az egész leírást, mire rájön az ember, hogy a Shift és vízszintes nyílbillentyűkkel tud a Konsole-on belüli shell ablakok között váltani.

– A Konsole még mindig nem képes a ⟨Delete⟩ és Shift-⟨Delete⟩ kombinációkat megkülönböztetni, pedig mind az XTerm és az Rxvt mindketten régóta találtak megoldást a problémára.

– A *Start menü/System/X terminal as root* egyszerűen nem működik, fölösleges volt berakni a menübe. Sőt, a Konsole *Session/New Root Console* menüpontja sem működik. Ezek helyett a *Star menü/KNOPPIX/Root shell*-t érdemes használni.

∗ Ugyan fel van rakva a KDevelop fejlesztőkörnyezet, de nem lehet benne dialógusdobozokat rajzolni, ezért semmivel sem kényelmesebb a használata, mint egy fapados szövegszerkesztőjé.

+ PostScript file-okat a Konqueror a Kghostview-val jeleníti meg.

– Ugyan megtalálható a Knoppix-ban a WINE nevű kitűnő, de még fejlesztés alatt álló Windows-emulátor, alapból nem ismeri fel a gépen levő Windows-partíciót (mert nincs be-mount-olva), ezért a kezdő felhasználó nem tudja használni.

* A különböző egyszerű szövegszerkesztők közül a *Start menü/Editors/NEdit* ajánlott.
+ A Knoppix része az OpenOffice irodai programcsomag 1.0.3-as verziója, és még számos irodai alkalmazást tartalmaz.
* A tálca ikonjai megnövekednek, ha föléjük visszük az egeret, de ez kikapcsolható (*KDE Control Center/Appearance/Panels/Enable icon zooming*)
* A start menü áttetszése a *KDE Control Center/Appearance/Style/Widget Style*-ban kapcsolható ki.
* A görbe ablakkeretek a *KDE Control Center/Appearance/Style/Window Decoration*-ben kapcsolhatók ki.
* Ha a *KDE Control Center/Appearance/Style/Theme Manager*-ben visszaváltunk a *Default* témára, akkor megszűnik az ablakok (pl. Konsole) áttetszősége.
+ Bár a Knoppix kevés támogatást nyújt a magyar nyelvhez, egyik régebbi verziójának létezik minden szempontból teljes magyarítása, Sulix néven, ami elérhető a `http://sulix.homelinux.net/` címről.
* Számos játékprogram található a CD-n, melyek többségének bonyolultsága és fejlettségi szintje a 80-as évek közepét idézi, de azért el lehet tölteni néhány órát mellettük.
+ A *Start menü/Settings/Desktop Settings Wizard*-ban a kinézetet érintő legtöbb döntést egy helyen meghozhatjuk.

Összességében elmondtható, hogy a Knoppix szerzői igencsak kitettek magukért, és a desktop kategóriában hátul kullogó Debian-ból egy friss, rengeteg hasznos programmal megpakolt, telepítés nélkül is jól használható egyfelhasználós rendszert állítottak össze, amely sok szempontból felveszi a versenyt egy Windows-os munkaállomással. A teljes integrálás feladatát azonban nekik sem sikerült megoldaniuk, rengeteg olyan művelet van, amit csak előzetes tudás birtokában, kerülő úton lehet végrehajtani.

6. Értékelés, kitekintés

A grafikus és szöveges felhasználói felületek manapság egymás mellett élnek, folyamatos csinosodás és fejlődés tapasztalható (főleg a grafikus esetében), az egyes konkrét felületek egymásra is hatnak, és a kinézetben keveredés alakul ki, megjelentek a felhasználó által választható skin-ek. A szépítés mellett a fejlesztők gyakran sajnálatosan kevés figyelmet fordítanak a használhatóságra, nincs egy átgondolt elképzelésük arról, hogyan lehet majd az általuk megtervezett felületen *hatékonyan* dolgozni, a napi munkát végezni, és nem vezetik rá a felhasználót erre a hatékony munkamenetbe. Ennek a fölösleges és időpazarló kattintássorozatok a következményei, melyek elvonják a felhasználó figyelmét a tényleges munkájáról. A szöveges felületeket manapság egy szűk, felkészült réteg használja, és pl. a shell-ek univerzális konfigurációs lehetőségeinek köszönhetően mindenki hatékonyabbá tudja tenni a saját munkáját. Megfigyelhető, hogy a grafikus modellek modelljei és irányelvei tiszták, pusztán a ma elérhető implementációk esetlegesek.

E sorok írója kialakított magának egy kényelmes szöveges és grafikus környezetet X11 alatt (fvwm2, XTerm és zsh felhasználásával), de a saját megoldását nem tudja bátran ajánlani, mert a lehetőségek kihasználásához, de főleg az apró módosításokhoz előismeretek szükségesek. A szerző azt szeretné, ha születne egy olyan integrált, könnyen konfigurálható és jól dokumentált grafikus felület, melyben a felhasználó szabadon megválaszthatná a widgetek méretét (helypazarlását), a csillogás szintjét, és dönthetne, hogy szüksége van-e panelra, start menüre stb. A Gnome és a KDE egyaránt alkalmas ilyen irányú továbbfejlesztésre, de ezek a környezetek óriási erőforrásigényűek, amit csak alapos újratervezéssel lehetne lefaragni. (Nem megoldás egy gyorsabb gép vásárlása, mert a grafikus felületnek akkor is azonnali válaszidőt kell biztosítania, amikor a gép erőforrásokat pazarló folyamatok nyomása alatt vergődik – és ilyen helyzet a gyorsabb géppel is előforulhat.) Az X11 világában a szerző integrált, de sokat tudó környezet kialakítására nem lát lehetőséget, mert túl sok fáradságot igényelne, és mire elkészülne, már megváltoznának a követelmények.

Vajon milyen koncepcionális változások várhatók? Nem valszínű, hogy a 3 dimenziós grafikus felületek népszerűvé válnak, mert a takarás miatt az információ szempontjából gyakran két dimenzióra csökken az ember látótere, továbbá nehezebb három dimenzióban közlekedni (gondoljunk az autó- és a repülőgépvezetés közti különbségre). Az orvosi műtéteken és a szimuláción kívül kevés olyan terület

képzelhető el, ahol a térbeli megjelenítés hatékonyabb beavatkozást tesz lehetővé a síkbelinél. Elképzelhető, hogy okosabbá válnak a grafikus felületek, szinte kitalálják a felhasználó gondolatait? Ez könnyen lehetséges, mindössze egy valósághű modellt kell alkotni arról, hogy a gyakorlott felhasználónak milyen igényei vannak. A megvalósítás feltétele az integrált desktop environment, melyet csak nagy tőkével vagy kisebb, versenyző fejlesztői csoport összefogásával lehet létrehozni. Mi fog változni, ha a képernyők felbontása kb. 600 DPI-re megnő? A zoom-olás és a görgetés szerepe csökkenni fog, a felhasználó szebb és tisztább képet fog látni, de az ablakok megszokott átlagmérete nem változik. Szükség van-e az egérnél pontosabb pozícionáló eszközre? Pontosabbra nem, hiszen az egeret egyetlen képpontal is odébb lehet vinni, gyorsabbra talán igen. A szem mozgását követő detektor sokat gyorsítana. A kattintás például határozott pislogással vagy kacsintással lenne megoldható. A tablet-ek megjelenésével az információbevitel módja megváltozhat, de mivel a tablet-en lassabban lehet gépelni, nem valószínű, hogy kiszorítják a billentyűzettel rendelkező személyi számítógépet.

A fentieket összefoglalva: a grafikus kijelző, a billentyűzet és az egér nyúltotta, évtizedek óta meglevő lehetőségeket a mai grafikus felületek nem használják ki teljesen, úgy, hogy a szépséget, a hatékonyságot és a gyors beletanulást egyaránt megvalósítanák. Ilyen irányú szoftverfejlődés igen kívánatos volna, de sajnos nem esik egybe a fejlesztők lehetőségeivel vagy gazdasági érdekeivel.