A nagy egyesített munkafelület

Hogyan vehetik rá a fejlesztők és a terjesztések a különféle programokat, hogy együtt jobban működjenek?

A Linux Journal weblapján 2002 decemberében megjelent egy cikkem (➲ http://www.linuxjournal.com/article/6476) a Red Hat 8.0 új Bluecurve nevű munkafelületéről, ami a Gnome és a KDE rendszerből egyaránt tartalmaz programokat. Számos visszajelzést kaptam, sőt néhány javaslat is érkezett, hogyan lehetne olyan alkalmazásokat írni, amelyek képesek együttműködni más eszközkészletekkel írt alkalmazásokkal, és minden munkafelületen jól működnének.

A kérdés

Az összevont munkafelület olyan szerkezet, aminek az elemei együttműködnek egymással, és soha nem kényszeríti arra a végfelhasználót, hogy ugyanazt a dolgot többször ismételgesse. A jelenlegi alapértelmezett megoldások (például a Gnome és a KDE) használata során gyakran ütközünk korlátokba, ha keverni szeretnénk az összetevőket.

A grafikus ügyfelek közötti valamennyi kapcsolat – az egyszerű húzd és ejtsd megoldástól kezdve az ikonok, menük, címek és más metaadatok kezeléséig, illetve általában az ablakszintű kapcsolattartásig – csak az ügyfelek és az ablakkezelők egy bizonyos alhalmazán működik kifogástalanul. Ugyanez vonatkozik a központilag kezelt és jól kinéző betűtípusokra.

No és mi a helyzet a nemzeti sajátságok kezelésével? Hány program létezik még, amit nem angol nyelven vagy billentyűzettel kínszenvedés használni?

A KDE és a Gnome két olyan alkalmazásgyűjtemény, ami úgy tűnik, sokat megold a felsorolt nehézségek közül. De valóban az eszközkészlet-központú, mindent vagy semmit alapú megközelítést lehet a legjobban fenntartani? Mi lesz, ha valaki még jobb eszközkészlettel áll elő? Mi történik a más eszközkészletekben írt alkalmazásokkal vagy az egész „best of breed” (a legjobb fajta) eszmeiséggel, ami olyan természetes a szabadprogram-világ felhasználói számára?

Más szavakkal: mi a legjobb módja annak a bizonyos legjobb alkalmazás a fejlesztésének, amit a kedvenc munkafelületünkhöz, mások kedvencéhez, illetve a jövő munkafelületeihez használunk?

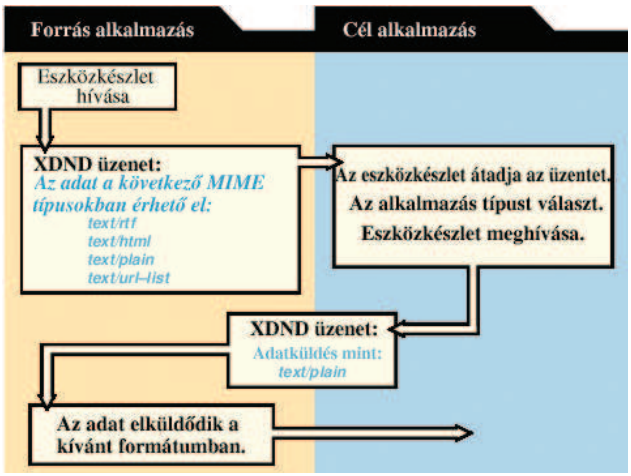
Megoldások

A kerék újrafeltalálása bolondságnak tűnhet, de sokkal kevésbé balgaság, mint olyan kereket fejleszteni, aminek újfajta utakra van szüksége. Amennyiben tehát létezik már húsz csevegőügyfél Linux alá, de úri kedvünk úgy tartja, készítsünk egy újabbat; azonban ne feledkezzünk meg arról sem, hogy együtt kell tudni működnie a már meglévőkkel.

Kezdjük józanul, és aknázzuk ki a helyes szabad szabványokat és protokollokat! Ezután ellenőrizzük, hogy a kiválasztott könyvtár vagy eszközkészlet szintén támogatja-e őket. Sok fejlesztő már rájött, hogy hosszú távon ez a nyerő megközelítés. ***Havoc Pennington***, Gnome-fejlesztő ezt írja: „Az alkalmazás és a futásidejű környezet közötti kapcsolattartásnak mindenképpen dokumentált eszközkészlet-független protokollokon és fájlformátumokon keresztül kell zajlania.” Lássuk, hogyan is fest mindez a gyakorlatban!

Fájlformátumok

A hivatali formátumok, de a strukturált adattárolás számos más területén is talán az egyik legígéretesebb dolog az Oasis csoport munkája. Az Oasis fájlformátumai az OpenOffice.org formátumaiból fejlődtek ki, így könnyebb idők következnek az



1. ábra XDND, az X húzd és ejtsd protokollja

OpenOffice.org fejlesztői számára. De minden program, az Emacstól a Koffice-on keresztül a jövő szövegszerkesztőjéig fel tudja használni.

Ezt az utat követve nincs szükség újabb könyvjelzőformátum készítésére. Az XBEL (XML Bookmark Exchange Language) olyan könyvjelző-csereformátum, amelyet a Galeon és a Konqueror is régebb óta használ.

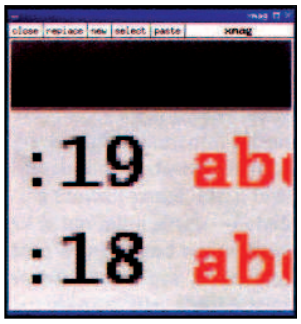
A csereformátum ötlete valamennyi beállításfájlra vonatkozik. A Unix és a Linux még mindig az eszközkészlet-megközelítést alkalmazza: nincsenek mamutok, csak sok kis darabka, amelyek kiválóan végzik a saját feladatukat. Ennek az az egyik hasznos következménye, hogy valamennyi azonos feladatot ellátó alkalmazás ugyanazt vagy egymáshoz nagyon hasonló értékkészletet ismer és állít be.

A csevegőügyfeles példánknál maradva: nem lenne túl okos külön beállításfájlokat készíteni Qt_chat_app, GTK_chat_app, és MyToolKit_chat_app és egyéb neveken a különféle eszközkészletekhez. Alapvető különbség van a fájl és annak előállítása, frissítése között. Használhatunk ugyan különféle szerkesztési módszereket (vi, vezérlőpult stb.), de hosszú távon csak az számít, hogy az összes létező csevegőügyfelünk egyetlen ***~/.chatrc*** fájlt használjon, és ne panaszkodjon, ha a beállításokat egy másik ügyfél megváltoztatja. A Usenet hírolvasó

például már régóta a szabványos ***.newsrc*** fájlt használja. Amíg az általunk készítendő alkalmazásfajtához még nem alakult ki szabvány, elfogadható egyezségnek tekinthető, ha az új programunk induláskor más hasonló ügyfelek beállításait tölti be, és azokat használja alapértelmezettként.
A MIME-típust fájlnév alapján a MIME-adatbázis végigkeresésével meg lehet határozni. Ezt szintén egységesíteni lehet, így minden munkafelület és program garantáltan azonos adatokat olvas be; ennek megvalósítási lehetőségeiről a
➲ http://freedesktop.org lapon vázlatos leírást találunk.

Grafikus felületek

Sokak szerint az X megfogja mutatni érettségét – és ilyenkor olyan ambiciózus, messzire mutató projektekre céloznak, mint a Fresco. Anélkül, hogy ilyen messzire mennénk, elmondhatjuk, sok mindent tehetünk annak érdekében, hogy az ügyfelek és az ablakkezelők bármilyen kombinációban helyesen együttműködjenek. Az ilyenfajta nehézségek megközelítéséhez kiindulási alapként a
➲ http://freedesktop.org honlapot javaslom, különös tekintettel a szabványok fejezetre. Emlékszünk még, mit is mondtunk? Az azonos típusú alkalmazásoknak azonos beállítási értékkészleten kell osztozniuk. A magszolgáltatásokra hivatkoztunk, de ugyanez a megközelítés igaz minden GUI-szintű beállításban az asztal összes eleme esetén. Az olyan jellemzőket, mint a háttérszín, a duplakattintás időzítése, minden program használja, mégis minden felhasználó csak egyszer és azonosan állítja be, függetlenül attól, hogy milyen eszközkészlet-keveréket használ éppen.



2. kép Élsimított szöveg

3. kép Szöveg hagyományos Xterm alatt

Az X Resource Managert nem készítették fel felhasználói beavatkozást igénylő kezelésre, s mivel a ***~/.Xdefaults*** tartalmát egyesíti más forrásokból származó adatokkal, nem éppen eszményi háttér GUI-beállítási eszköznek. Inkább tekintsük meg az Xsetting-meghatározásokat, amiket ennek és más nehézségeknek a legyőzésére készítettek.
Lépjünk tovább! Mi következik, miután az egyes alkalmazásosztályokat, a megjelenésüket, illetve a helyi felhasználók cselekedeteire adott válaszaikat beállítottuk? A más alkalmazásokkal és az ablakkezelővel történő grafikus kapcsolattartás kérdése. Már most is lehetőségünk van szövegeket húzni át egyik ablakból a másikba, legalábbis elméletileg, amennyiben az összes résztvevő követi az XDND protokollt. Jelenleg nemcsak a GTK+ és a Qt, de még a parancsfájlközpontú eszközkészletek is támogatják, mint a Tk és a Perl/Tk, illetve az olyan viszonylag kívülállók, mint a FOX.
Az *1. ábra* szemlélteti, hogyan működik az XDND. Amikor az alkalmazásnak valamit át kell küldenie egy másiknak, kiadja a megfelelő eszközkészlethívást. Az eszközkészlet az XDND protokollon keresztül mindent elküld a másik eszközkészletnek, majd a válasz ugyanezen az úton kerül vissza a programhoz.
A forrás az eszközkészleténekmegadja az összes általa támogatott MIME-típust, amik lehetnek különféle szöveges formátumok, képek vagy fájlnevek. A célnak meg kell mondania, hogy melyik formátumot ismerte fel az adott listából. Így a formázott szöveget két szövegszerkesztő között is átadhatjuk, anélkül, hogy a karakterkészletet, a színeket és a többi formázást elvesztenénk. Ugyanakkor ha egy hasábot átemelünk mondjuk a KOffice-ból a vi alá, akkor a formázás elvész, de megmarad a tartalom.
Amikor a húzd és ejtsd nem működik, a hiba oka csaknem mindig az alkalmazás XS-ében keresendő, ami nem adta meg az összes MIME-típust, csak saját formákat alkalmazott, vagy hibásan használja az eszközkészletet, esetleg a protokollt. Ez gyakorlatilag azt jelenti, hogy a hibajelentést az alkalmazáshoz kell benyújtanunk, és nem az eszközkészlethez vagy az XDND-hez.
Az X-ügyfelek közti, illetve az ügyfelek és az ablakkezelő közötti egyéb kapcsolatoknak az ICCCM szabványt (Inter-Client Communication Conventions Manual), illetve az Extended Window Manager Hintset, röviden EWMH-t, a korábban NetWM néven ismert leírást kell követniük. Megjegyezzük, hogy az ICCCM még a jó öreg X vágólap részletes ismertetését is tartalmazza. Kezelésükről részletes magyarázatot kaphatunk ***Keith Packard***-tól és ***Jamie Zawinski***-től.
A második ablakkezelési szabvány az ICCCM-re épül. Az összes olyan ablakkezelési képességet kezeli, amit az elődje nem tartalmazott, mivel azok az ICCCM létrehozása után jelentek meg. Az EWMH eredetileg az akkori Gnome ablakkezelő leírás helyettesítésére készült, de ma már (helyesen) bármilyen munkafelületen megvalósítható és támogatott. A Gnome 2 és a KDE 3 egyaránt támogatja az EWMH-t, így minden alkalmazás, ami használja, várhatóan mindkét környezetben, illetve azok elemeivel szépen fut majd. Azoknak a fejlesztőknek, akik az EWMH-ról ennél a gyors áttekintésnél pontosabb leírásra vágynak, érdemes belenézniük a KDE ***netwm.h*** állományába.

Betűtípusok és kódolás

Mindent elmondtunk már, ami a nagy egyesített munkafelülethez szükséges lehet? Nyilvánvalóan nem. Gondoljunk csak a Red Hat 8.0-ra. Szinte mindenki, még azok is, akiket a hideg ráz ki a Gnome/KDE keverésétől, egyetértenek abban, hogy legalább a betűk sokkal jobban néznek ki. Mi is könnyen becsempészhetjük a saját terjesztésünkbe ezt a varázslatot.
Először is, a fejlődés az élsimítás érdeme, ami szebbé varázsolja a karaktereket, pixeleket szúrva be a megfelelő helyekre. A *2.* és *3. kép* ugyanazt a szöveget mutatja be egy szabványos `xterm` és egy élsimításos változatban.
Másodszor a Red Hat 8.0 az első vezető terjesztés, ami a lehető legtöbb UTF-8 kódolást használta (erről még később lesz szó), illetve beépítette az `xft2` és ***fontconfig*** könyvtárak által nyújtott ügyféloldali betűtípus-kezelő rendszert.
A ***fontconfig*** képes kitalálni, hogy milyen betűtípusok érhetők el a rendszeren és hogy milyen dokumentumhoz melyik felel meg a leginkább. Ha ez megvan, az `xft2` mondja meg az X-kiszolgálónak, hogy mit kell kirajzolnia. Mindkét könyvtárnak kapcsolatban kell állnia a FreeType vagy egy azzal egyenértékű raszterizáló motorral. Ez a következőket jelenti:

- Mostantól nincs szükség betűtípus-kiszolgálókra.
- Az új betűtípusok telepítése (akár rendszergazdai jelszó nélkül is) sokkal egyszerűbb lett.
- Minden alkalmazás, ami a ***fontconfigon*** alapul, az összes betűtípus-beállítást egyazon XML-fájl(ok)ból olvassa ki, amit bármilyen felülettel szerkeszthetünk.

4. kép Emacs Red Hat 8.0 alatt

- A betűtípus-kezelés mostantól (bármely alkalmazásban) e két könyvtár sebességével történik, és nem magának az X-nek a sebességével.
- Mivel a ***fontconfig***-nak nincs szüksége se `xft2`-re, sem más X-szel kapcsolatos elemre, bármibe beágyazható, ami betűtípusokat kezel – akár nyomtatómeghajtókba vagy könyvtárakba is. A ***libgnomeprint22*** pontosan ezt is teszi.

Az xft2/fontconfig rendszer másik ügyes megoldása, hogy nem mindent vagy semmit alapú. Békésen megfér a hagyományos betűtípus-kiszolgálók mellett, amik a régebbi alkalmazásokhoz esetleg szükségesek lehetnek. Ez történt a Red Hat 8.0 esetében is. Az `xft2` pedig az új és a régi XFree86-kiszolgálókkal is beszélgetni tud.

Előfordul azonban, hogy a szép szimbólumok rajzolgatása még nem elegendő, és az alkalmazásunknak fejlett szövegmegjelenítésre is szüksége van. Ilyenkor kerül a képbe a ***fontconfig***-on alapuló Pango program. A Pango szintén a többé-kevésbé monolitikus Gnome munkafelületen született, de jelenleg már bármilyen környezetben futtathatóra tervezik.

Egy utolsó gondolat a betűtípusokról: a szebb rajzolat jó dolog, de ha kizárólag számjegyekhez és az angol ábécéhez használhatjuk őket, akkor az egész nem igazán érte meg, nemde?

Ha a kelet-európai, afrikai vagy ázsiai nyelveket nézzük, az ASCII még a „Szia, Világ!” megfelelőjét sem tudja visszaadni. A fejlesztő oldaláról nézve ez annyit jelent, hogy az alkalmazásokat az első naptól kezdve többnyelvűre kell tervezni, és minden, már létező programot ellenőrizni kell, hogy biztosan működik-e. Ez nem csupán egyszerű javaslat. Ha valaki azt hiszi, hogy biztonságban van, mert csak ASCII-t használ és csak néhány parancsfájlt alkotott, az gondolja végig még egyszer a dolgot. Amikor a legközelebb fejlesztünk, és a kódunk egy nemzetköziesített parancsértelmezőben, terminálban vagy ablakkezelőben indul majd el, könnyen lerobbanhat. Perlben íródott, véletlen aláírásokhoz készült programom pontosan emiatt állt le Red Hat 8.0 alatt, és csaknem három hónapig három különböző listán senki sem tudott használható megoldást javasolni.

A Linux alatt lassan szabvánnyá váló karakterkódolás az Unicode UTF-8 (lásd ***Reuven M. Lerner*** Unicode című cikkét a Linuxvilág márciusi számában). Jó hír, hogy ez már a bolygón található valamennyi jelet képes rendszerbe foglalni, így semmilyen más kódolásra nincs szükségünk. A *4. képen* a Red Hat 8.0-hoz kapott Emacs látható, amint az összes szimbólumot és karaktert megjeleníti. Rossz hír viszont, hogy mivel a nem ASCII-karakterek több mint 8 bitet használnak, illetve néha a képernyőn is nagyobb képterület igényelnek, számos mélyen beivódott tézis, például az „1 karakter = 1 byte” egyenlet, egyszerűen megszűnik. Témába vágó forrásmű (a Linux Unicode HOWTO-n kívül) az „UTF-8 soft and hard conversion” mini útmutató és az `UTF8_STRING` szerkezet, aminek célja megőrizni az X vágólap rendszerét az UTF-8 világban. Magasabb szinten az ➲ http://Openi18n.org feladata az operációs rendszertől független, nemzetköziesített programozás.

Menük és ikonok

Mi kellhet még a szép GUI-hoz? Menük és ikonok.

A ➲ http://freedesktop.org-on már most megtalálható a munkafelületek alapszintű szabványa, ami környezettől függetlenül megadja, hogyan kell felépíteni a menüket és elindítani az alkalmazásokat. Igaz, megvannak a maga korlátai, például annak a közös helynek a hiánya, ahová a ***.desktop***-fájlokat be lehetne tenni; illetve a menük beégetése, ami abból a tényből fakad, hogy egyszerűen a merevlemezen található fájlok elhelyezkedését tükrözik. Ezeket a hátrányokat kiküszöbölendő a szabványhoz már készül a virtuáliskönyvtár-bővítmény. Elérhető egy hasonló témájú, másik meghatározás is, ami az ikonhelyeket és a témaválasztást szabványosítja.

A telepítés egyszerűsége

Ha a forrást is terjesztjük, nyilván mindenki lefordíthatja és telepítheti magának a programunkat. De miért nehezítsük meg mások életét azzal, hogy nekik kelljen rájönniük, miért is nem találja az a program a feltelepített programkönyvtárakat? Miért égessük be a dolgokat, ha azok így csak egyetlen terjesztésen fognak működni? Bármilyen alkalmazást is szeretnénk írni, az LSB csoport Linux fájlrendszerszerkezete jól jöhet ilyenkor.

Összegzés

Semmi bajom a tiszta KDE vagy tiszta Gnome rendszerekkel. Mindössze csak azt remélem, hogy a jövő alkalmazásaiban egyszerűbb lesz bármilyen programkombinációból a saját környezetünket összeállítani, a valós szolgáltatások és a teljesítmény feláldozása nélkül is. Az itt leírt módszerek és eszközök sokat segíthetnek egy ilyen alkalmazás megírásakor, fejlesztőiket köszönet illeti.

Köszönetnyilvánítás

Köszönetet szeretnék mondani még ***Havoc Penningtonnak***, ***Keith Packardnak***, a kde-devel lista tagjainak és mindenkinek, aki válaszolt a linuxjournal.com weblapra, és ezzel hozzájárult e cikk megszületéséhez.

Linux Journal 2003. április, 108. szám

A kapcsolódó címek a CD Magazin/Munkafelület könyvtárában találhatóak.

Marco Fioretti
Alkatrész-rendszermérnök, aki sokat foglalkozik a szabad programokkal az EDA-felület és hatékony asztali rendszerek formájában. Marco családjával Rómában él.