Windows-fejlesztés Linux alatt

Fordítsuk és teszteljük a projekt Linux és Microsoft Windows változatát újraindítás nélkül! A MinGW és a Wine ingyenes eszközökkel a Win32 felületet akár több felületen működő API-nak is nevezhetjük.

A legtöbb ***Linux Journal*** olvasóhoz hasonlóan én is elvakult rajongója vagyok minden Linux, GNU és nyílt forráskódú dolognak. Saját gépeimen Linuxot használok, ezeken programozom, játszom és térítek másokat, amikor csak lehet. A létező programozási munkák nagy része viszont olyan alkalmazások írását jelenti, amelyek a Washington állam-béli Redmondból származó operációs rendszeren futnak. A munkám révén kénytelen voltam néhány kisebb alkalmazást írni Microsoft Windows felületre. Mivel a futás sebessége lényeges volt, C-ben akartam megírni a programokat, közvetlenül használva a Win32 alkalmazásprogramozói felületet. Felmerült bennem, hogy ha olyan szabványos nyelvet használok, mint a C, akkor fejleszthetek a szép és kényelmes Linux birodalmamban.
Ez a cikk egy rövid útmutató ahhoz, hogyan lehet windowsos alkalmazást fejleszteni linuxos környezetben. Rövid bevezetőt adok a windowsos programozásból, majd lépésről lépésre végigmegyünk egy példaprogram fordításán és tesztelésén. Szó lesz még a Wine-ról, hogy hogyan lehet a segítségével leegyszerűsíteni a Windows forráskód átemelését a Linuxba.

Win32 programozás

Azoknak, akik a UNIX-féle folyamat-elvonatkoztatás egészséges táplálékán nőttünk fel, a Windows-modell egyenesen eretneknek tűnhet. A Windows-modell egy időosztásos, többfeladatos, többszálú, üzenetátadásos operációs rendszer. Most csak az NT-re és a leszármazottaira, az 2000-re és az XP-re szorítkozom. Az operációs rendszer minden folyamatot egy szálnak tekint. Ez a folyamat-összefüggést valamivel könnyebbé teszi, mint a hagyományos nehézsúlyú folyamat-modell, amely a UNIX-hoz hasonló operációs rendszerekben használatos. Ennek a „minden egy szál" modellnek a következtében viszont minden egy globális memóriacímtérben van. A megfelelő jogosultságokkal és a megfelelő cím birtokában az egyik program piszkálhatja a másik program bitjeit. A másik következmény, hogy a mag által létrehozott adatszerkezetek nem rögzített memóriacímeken vannak. Ez azt jelenti, hogy a felhasználó programnak kell lefoglalni a kapcsolódó memóriát, mielőtt bármilyen globális adatszerkezetet használna, például grafikus környezetet. Arról sem szabad megfeledkezni, hogy felszabadítsuk ezeket a szerkezeteket, miután végeztünk, különben memóriatöredezettséget okozhatnak. Az *1. lista*, amely a Linux Journal FTP oldalán elérhető, egy alap „Hello World" program. Nagyrészt szabványos, csak a `switch` utasításon belüli rész érdekes igazán. Egy alap programhoz képest elég soknak tűnik a kód, de épp ez a baj az alacsony szintű API használatával. Jó összehasonlítás lenne erre a Linuxban, ha az Xt segítségével akarnánk kódot írni az X-hez.
A `main()` függvény helyett a Windows GUI (grafikus felhasználói felület) alatt futó program `WinMain()`-nel kezdődik. Ehhez a függvényhez hozzátartozik, hogy a program végzi el az egész kezdeti értékadást. Ennek a kezdeti értékadásnak része a windowosztály megadása a főablakhoz, és egy visszahívandó (callback) függvény hozzárendelése. Ezután hozzuk létre a főablakot és jelenítsük meg az asztalon. Ekkor a vezérlés az üzenetkezelő ciklushoz kerül, és a visszahívandó függvény feldolgozza a főablakhoz küldött üzeneteket.
A ➲ http://www.winprog.org-on elérhető egy jó és gyors bevezető a Windows programok írásába.
A webhely szerkesztői egy jó *faq*-t és egy egész jó, az összes alapkérdést átfogó segédletet kínálnak. Természetesen, a Windows programozás bibliája a vastag „***Windows Programming***" könyv, ***Charles Petzold***-tól. Ha ebben a kötetben nem találjuk, amire szükség van, még mindig elég tekintélyes méretű ahhoz, hogy kiverjük vele az információt a legközelebbi Windows-guruból.

Keresztfordítás

A GCC-ben az az egyik bámulatos dolog, hogy annyi különböző felületre és operációs rendszerre átültették már. Ennek nagy előnye, hogy olyan binárisokat fordíthatunk egy felületen, amelyek teljesen más felületen futnak. Én rendszeresen fordítok Solaris vagy Windows binárisokat a linuxos hordozható gépemen. Ez hihetetlen előny, mert lehetővé teszi, hogy a fejlesztés egy ismerős, kényelmes környezetben történjen.
A legtisztább módon úgy kezdhetünk hozzá, hogy visszamegyünk a forráshoz (lásd források). Így pontosan olyan beállításokkal és olyan felületen fordíthatjuk a kódot, ahogy szeretnénk. Szerencsére, ezt a munkát már elvégezték he-

lyettünk. A MinGW Project kedves munkatársai fenntartanak egy GCC-változatot windowsos binárisok fordítására. Ez tartalmazza az összes kapcsolódó állományt, például a fejállományokat (headers). Itt megtalálhatók a források, a bináris archívumokkal együtt. A programokat RPM-alapú és deb-alapú terjesztések számára is becsomagolták. Ha Debiant használunk, az `apt-get` segítségével letölthetjük a ***mingw32*** vagy a ***mingw32-runtime*** csomagot. Ha testing vagy unstable (Sarge, SID) változaton dolgozunk, akkor a ***mingw32-binutils***-t is le kell tölteni.
A GCC legtöbb fordítási lehetősége megtalálható itt a ***MinGW***-ben, néhány ráadással. Ha egyszerűen csak lefordítunk egy programot, különleges lehetőségek nélkül, akkor futtatható a konzolról, például ha egy kicsi, egyszerű programot szeretnénk írni, amelyikhez nem kell GUI. Mivel ez Windows és GUI programot akarunk, felkészítjük a kódunkat a fent leírtak szerint, és a fordítási parancshoz hozzáadjuk a `-mwindows` kapcsolót. Ez létrehozza egy szabvány Windows végrehajtható állomány fordításához szükséges makrókat és könyvtári hivatkozásokat. Ha úgy döntünk, hogy bonyolultabb Windows programot írunk, amely más Windows-szolgáltatásokat is használ, akkor pontosan meg kell adni a könyvtárakban a szükséges hivatkozásokat. A Windowsban meghatározhatjuk a program erőforrásait, például menüket, bitképeket, szöveget és más elemeket. Ezeket az erőforrásokat egy külön állományban tároljuk és külön le kell fordítani, mielőtt a végrehajtandó állományhoz kapcsoljuk. Ez a feladat a `mingw-windres` programra marad, amely létrehoz egy objektumállományt, amit később a végrehajtható állományhoz kapcsolhatunk. Az 1. listában látható egyszerű példaprogram fordításához a következő parancsot használjuk:

```
mingw-gcc -o example1.exe example.c -mwindows
```

Cseréljük ki a `mingw-gcc` parancsot bármire, ami az adott csomaghoz jó, mint ***Compiler executable***. Íme, van egy életképes Windows programunk. Ilyen egyszerű!

Hibakeresés a Winenal

A Wine a másik nagy adomány azoknak a fejlesztőknek, akiknek Windows-felületen kell programokat írniuk. A fejlesztők elképesztő mennyiségű munkát fektettek a Wine-ba. Ez a nagyszerű program lehetővé teszi, hogy windowsos programokat futtassunk Linux alatt. Ennek az az eredménye, hogy a frissen fordított programunkat lefuttathatjuk, és megnézhetjük, hogy valóban működik-e, ahogy beharangoztuk. Ehhez adjuk ki a `wine example1.exe` parancsot, és látnunk kell az ablakot megjelenni az asztalon. A Wine indításakor megvan a lehetőségünk, hogy az ablakokat a saját ablakkezelő felügyelje, és ne a saját asztalukon jelenjenek meg. Ez befolyásolja azt, hogy néz ki az ablak futtatáskor. Ha nem voltunk elég ügyesek ahhoz, hogy hibátlanul gépeljük be a programot, akkor hibakeresésre lesz szükség, hogy kiderüljön, mi nem jó. A Wine ebben nagy segítség lehet.
A `-debugmsg [debugchannel]` lehetőség a Wine egy vagy több hibakereső csatornájának eredményét adja kimenetként. Példa a lehetséges hibakereső csatornákra:

- `relay`: naplóüzenetet ír, akárhányszor egy Win32 függvény meghívódik.
- `win`: nyomon követi a Windows-üzeneteket.
- `all`: nyomon követi az összes üzenetet.

Az `all` lehetőséget ne használjuk, csak ha valóban szükség van rá, mert a kimenet mennyisége könnyen kifoghat legrészletmániásabb programozón is. A Wine oldalon megtalálható a lehetséges hibakereső csatornák teljes felsorolása.

Eredeti változat fordítása Linuxra

Most van egy csodálatos, működő, hibamentes programunk, amely Windows alatt fut. Tekintve, hogy az egész munkát Linux alatt végeztük, nem lenne nagyszerű, ha a program Linux alatt is futna? A Wine Project kedves munkatársai ismét a segítségünkre sietnek. A projekt egyik része tartalmazza a ***winelib*** könyvtárat, amely a Windows forráskód számára biztosítja a linuxos illesztőfelületet. Hogy használhassuk ezt a szolgáltatást, telepíteni kell a ***wine-devel*** csomagot a terjesztéshez. Ha forrásból telepítettünk, a szükséges fájloknak már hozzáférhetőnek kell lenniük.
A ***wine-devel*** csomag egy `winemaker` nevű Perl parancsállományt tartalmaz, amelynek az a feladata, hogy végigmenjen a forrásfájlokon és könyvtárakon, és végrehajtsa a Linux ***winelib*** könyvtárában történő helyes fordításhoz szükséges változtatásokat. Olyan dolgokat ellenőriz, mint az állománynevek nagy- vagy kisbetűs írásmódja és a sorvég karakterek. Ráadásként a fájlok elérési útjában a fordított perjeleket kicseréli hagyományos perjelekre, és sok más dolgot elvégez. Alaphelyzetben biztonsági másolatot készít azokról az állományokról, amelyeket meg kell változtatnia. Átalakítja a projektet winelibbé, miden változtatást magától elvégezve. A fordításhoz egyszerűen futtassuk a következő parancsokat, ami által linuxos végrehajtható állományt hozunk létre.

```
winemaker .
./configure --with-wine=/elérési/út/wine
make
```

A fenti pont szándékosan van ott. Megadjuk az elérési utat, amelyben a `winemaker` megtalálja a vizsgálandó forrásfájlokat. Itt a fájlok az aktuális könyvtárban vannak.
A mi esetünkben a példában nincsenek projektfájlok, és ezt a `winemaker` problémaként érzékeli. A szükséges lépéseket egyszerűen elvégezhetjük kézzel. A program fordítására a `mingw-gcc` végrehajtása helyett `winegcc`-t használjuk, pontosan ugyanazokkal az értékekkel. Ez létrehoz egy `.so` végződésű állományt, és egy héjprogramot, amely a program futását felügyeli. Most már lefordítottuk a Windows forráskódot, és működik Linux alatt.

Következtetés

Remélem, sikerült meggyőznöm néhány windowsos fejlesztőt arról, hogy hatékonyan dolgozhatnak Linux alól. A GCC-vel, ami lehetővé teszi windowsos végrehajtható állományok fordítását és a Wine-nal, amely sok támogatást nyújt a futtatáshoz és a hibakereséshez, legtöbbször nincs oka, hogy egyáltalán elindítsuk a Windowst. Az egyetlen ok az lehet, ha a kedvenc fejlesztőkörnyezetünk nem fut rendesen Wine alatt. Szerencsére egyre több rendszer fut hibátlanul Wine alatt is.

Linux Journal 2004. július, 123. szám

Joey Bernard