A szerkesztők és az Emacs (1. rész)

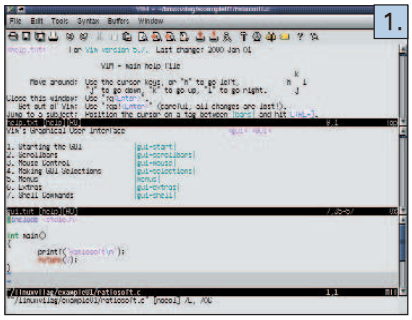
Ezt az írást nem a Linux-rendszert és a programozást jól ismerőknek szánom, hanem azoknak, akik mindkettővel most ismerkednek.

Elsőként a forráskód megírásához használható szerkesztőket mutatom be a teljesség igénye nélkül, majd az Emacs fejlesztőkörnyezettel foglalkozom hosszabban.

Szövegszerkesztők

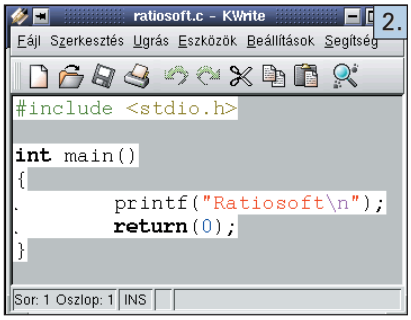
A programokat először meg kell írni, amihez kell egy nekünk tetsző egyszerű vagy kevésbé egyszerű szövegszerkesztő, amely ASCII formátumú állományt hoz létre. Ebből következik, hogy haszontalan lenne az OpenOffice nagy tudású szerkesztőjét használni, mert fölöslegesen sok helyet foglal el a memóriában, és semmiképpen sem tudjuk kihasználni a képességeit, hiszen a programírásban nem segítenek a kifinomult formázási lehetőségek, nem célszerű sokféle betűkészletet használni, és számos programozót zavar, ha változó szélességű és nagyságú betűket lát maga előtt bogarászás közben. Ne feledjük, hogy bármilyen szépre is formázzuk meg forráskódunkat, a formázó karakterek abban a pillanatban nyom nélkül eltűnnek, amikor forráskódunkat ASCII formátumban kimentjük, a fordítók pedig csak a formázatlan állományokat hajlandók elfogadni!

A fentiekből következik, hogy a merészebbek programozásra is használhatják a legendás hírű ***vi*** szerkesztőt. Erről a kezdőknek azt illik tudni, hogy a kellően tájékozottak „viáj”-nak ejtik, és hogy a `:q!` paranccsal lehet kilépni belőle. Ez nem segít mindig, ilyenkor nyomjuk meg az Esc billentyűt, és próbálkozzunk újra. Ha már fel vagyunk vértezve ennyi tudással, akkor legalább nem kell újraindítani az egész rendszert, hogy visszakaphassuk a parancssor készenléti jelét, ahogy azt egyik ismerősöm tette a `vi`-t próbálgatva. A mindenre elszántak próbálkozhatnak a `vi` többalakos változatával, az `xvi` (ejtsd: „eksz-vi-áj”) szerkesztővel, vagy az `xvi`-nál is fejlettebb `vim`-mel. A `vim` a „vi improved” (tökéletesített `vi`) szavak rövidítése. A `vi`-nak az elmúlt évtizedekben számos mutációja született, és ha telepítve vannak a gépünkön, kipróbálhatjuk őket. Írjuk be a parancssoron az `ed`, `ex`, `gex`, `vi`, `gvi`, `view`, `gview`, `vim`, `gvim`, `rvim`, `rview`, `rgvim` vagy `rgview`, `elvis`, `nvi` parancsokat és az olvasni vagy szerkeszteni kívánt állomány nevét! Ezek a parancsok többnyire ugyanazt a végrehajtható állományt indítják el, de különböző bővítményekkel vagy éppen korlátozásokkal. Például a `view` vagy a view-ra végződő parancsok használatakor a megnyitott fájlt csak olvasni lehet, a parancsok elején lévő r betű pedig a „restriction”, azaz megszorítás szóra utal. A korlátozott parancsok valamilyen

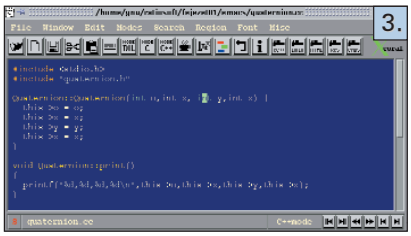

1.

módon szűkítik a `vi` lehetőségeit, például nem indíthatunk parancsokat a `vi`-ból, vagy nem függeszthetjük fel ideiglenesen a működését. A `gvim` (*1. kép*) a Windowsban megszokott menüket és ikonokat varázsolja elénk, és viszonylag jó leírással bír. Első pillanattól otthon fogjuk érezni magunkat benne, ha a telepítés után sikerül elindítanunk.

Ha nem, próbáljuk meg a *~/.gvimrc* fájlt testreszabni. A `vi` általában nem alkalmas bináris állományok szerkesztésére, ehhez inkább a `hex`, `vche`, `vche-nc`, `vche-raw`, `khexedit` vagy a `ghex` programokat használjuk. Szintén kéznél van a Midnight Commander belső fájlnézegetője, amely hexadecimális szerkesztő is egyben. A `vimtutor` program végigvezet bennünket a legfontosabb `vi`-parancsokon. Többek közt megtudhatjuk, hogy mely két üzemmód létezik a `vi`-ban, hogyan indíthatjuk el, hogyan mozgathatjuk a kurzort, hogyan szerkeszthetjük a fájlt, és miképpen léphetünk ki belőle. A `pico` szintén parancssori szövegszerkesztő, de könnyebb megtanulni, mint a `vi`-t. Ezt már az is mutatja, hogy mindjárt induláskor kiírja, hogyan lehet kilépni belőle. Megemlítem még a ***Joe*** szerkesztőt, amit ötféle üzemmódban használhatunk. Ha a `joe` helyett a `jpico` parancsot ütjük be, akkor a Joe a most említett Pico szerkesztőt utánozza,



2.

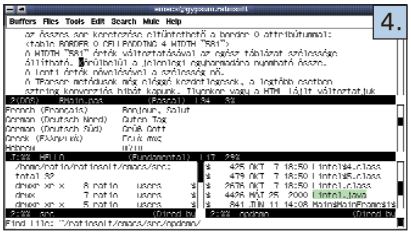

3.

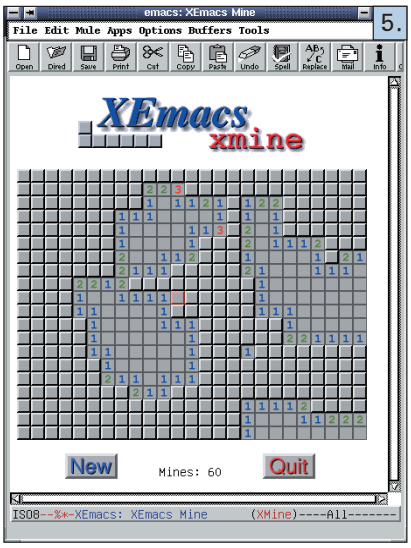
ha a `jstar` parancsot, akkor a hajdan jól ismert WordStar szövegszerkesztőt. A fenti programok mindegyike igen nagy tudású, de használatukhoz némi tapasztalatra van szükség. Az `xedit`, `kedit` és a `gedit` a Windows felől érkezők számára minden bizonnyal barátságosabbnak fognak tűnni, mint parancssori elődeik, de azt is rögtön tapasztalhatjuk, hogy ezek sem tudnak sokkal többet, mint a `notepad.exe`.

Az Emacs

A Linux-programozással foglalkozó szakkönyvek szinte kivétel nélkül a *GNU Emacs* fejlesztőkörnyezetet ajánlják. Figyeljük meg, hogy immár nem a „szerkesztő” szót használtam, hanem az IDE (Integrated Development Environment) mozaikszóra utaltam, ami magyarul összetett fejlesztőkörnyezetet jelent. Mégse gondoljunk azonban a *Linuxvilág* augusztusi számában ismertetett KDevelophoz vagy a Borland Kylixhoz hasonló RAD (Rapid Applica-

tion Development, azaz gyors alkalmazásfejlesztés) eszközökre. Az Emacs a Unix-hagyományokban gyökeredzik, nehézkes és barokkosan bonyolult. Némelyek szerint az Emacs gonosz, mások viszont három részre osztják a világban élő embereket: azokra, akik Emacsot használnak; olyanokra, akik inkább a


4.



5.

vi-t szeretik; és mindenki másra. Vannak, akik azt mondják, hogy az Emacs egy jó operációs rendszer, bár elismerik, hogy a Unixban több program van. Bonyolultságából következik, hogy hosszú időbe telik, amíg otthonosan mozoghatunk benne, és inkább azok számára ajánlom, akik naponta fogják használni, hiszen elég pár hetes kihagyás, és máris jó néhányat elfelejthetünk a számtalan billentyűkombinációból.

Az Emacs név a ***Richard M. Stallman*** által a TECO szerkesztőhöz írt „editing macros” (szerkesztőmakrók) szavak betűiből alkotott mozaikszó. Az Emacs történetét elolvashatjuk a ➲ http://www.gnu.org/philosophy/stallman-kth.html (a fájl másolata megtalálható a 24. CD Magazin/Emacs könyvtárában). A `vi`-hoz hasonlóan az Emacsnak (*4. kép*) is több változata létezik, például a ***Xemacs*** (*5. kép*), ami egy eszköztárral is rendelkezik. Az utóbbit korábban ***Lucid*** (azaz világos, érthető) ***Emacs***-nak nevezték.

Mindkét Emacs-fajta elfogadott, és gyakran együtt hivatkoznak rájuk az Emacson szóval. Ahogy az Emacs bejelentkező ablakában olvashatjuk, a GNU Emacs az egyik alkotóeleme a Linux-alapú GNU-rendszereknek, és mint ilyet egyetlen Linux-terjesztésből sem illik kihagyni. Akik GNU-programokat akarnak írni, azoknak otthonosan kell mozogniuk az Emacsban, mint ahogyan a rendszergazdáknak is ismerniük kell a `vi`-t, ami kötelezően ott van minden Unix-típusú rendszerben, és amit a legvadabb rendszerösszeomlások idején is el lehet indítani. Mindkét program elengedhetetlen kelléke a Unix-kultúrának, a Linux-életérzésnek és a GNU-mozgalomnak.

Ahogy a képeken is látjuk, az Emacs egyszerre több fájlt is meg tud nyitni – akár ugyanazon keretben több részre osztva fel a munkaterületet. Számos keretet vagy ablakot tarthatunk nyitva egyidejűleg, futtatható állományokat és képeket nézhetünk meg vele. Ha az Emacs nem támogatott képformátumot talál, akkor külső képnézőt (például ImageMagick) hív segítségül. Programozás közben intézhetjük levelezésünket, honlapokat látogathatunk meg és tölthetünk le akár PostScript formátumban is anélkül, hogy kilépnénk az Emacsból. Amikor elfáradunk, néhány beépített játékkal játszhatunk.

Az Emacsot nem a manapság igen divatos C vagy C++ nyelven írták és írják, hanem a Lisp nyelvet használják erre a feladatra, annak is egy különleges fajtáját, az Elispet, más néven Emacs Lispet. Az Elisp teljes értékű programozási környezet, amivel szöveget és fájlokat kezelhetünk, hálózati alkalmazásokat vagy új felhasználói felületeket építhetünk fel az Emacson belüli használatra. Az érdeklődők kipróbálhatják még a Jonathan Saját Emacs Változata (Jonathan's Own Version of Emacs) programot vagy annak menüsített, XJove nevű kiadását, ha beírják a `jove` vagy `xjove` parancsokat. Az `uemacs` vagy más néven MicroEmacs kis teljesítményű gépekre készült, és a 4.0-s változathoz maga ***Linus Torvalds*** írt javításokat. Utóbbit az `em` paranccsal indíthatjuk.

Az Emacs logikája

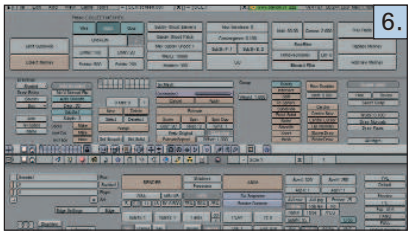
Talán igazságtalanul bántam az Emacscsal, amikor azt mondtam róla, hogy nehézkes és bonyolult, hiszen már rövid használat után beláthatjuk, hogy a fejlesztők egységes és igen egyszerű elvek alapján építették fel a programot. Ezeket az elveket így foglalhatnánk össze:

Ha a program használata közben új szempontok merülnek fel, amik megkönnyíthetik a mindennapi munkát, akkor meg kell valósítani azokat. Az új feladatok ellátásához egy vagy több Elisp-függvényt kell megírni, és ezeket a függvényeket a felhasználók számára hozzáférhetővé kell tenni, azaz meg kell adni számukra a lehetőséget, hogy programozás nélkül meghívhassák őket.

Mivel Richard Stallman és társai Unix-környezetben nevelkedtek, vérükké vált az a szemlélet, hogy egy program csak egy dolgot csináljon, de azt felettébb jól. Követelmény volt a Unixban az is, hogy ezek a programok tetszés szerint összefűzhetők legyenek. S valóban, az Emacsban ott van a mini átmeneti tár, ami lényegében egy miniatűr héj, ahová parancsokat gépelhetünk be. A mini átmeneti tárhoz hasonló parancssori beviteli eszközök eltűntek a mai szövegszerkesztőkből, ha egyáltalán voltak bennük, de a táblázatkezelőkben a szerkesztőlécek még most is megtalálhatók, annak ellenére, hogy az adatokat közvetlenül a cellákba is beírhatnánk. A Unix-követelmények szerint az Emacs-parancsok szintén kötegelhetők, és rendszerint csak egy dolgot tesznek, de azt felettébb jól, hiszen a közel két évtizedes fejlesztés következtében az Emacs igen jól átgondolt és hibamentes lett. Azt is tudjuk, hogy a programozó nem szeret sokat gépelni, annak ellenére, hogy a billentyűzetet mesteri módon tudja kezelni. Magától adódott tehát a felismerés, hogy az egyre szaporodó parancsokat billentyűkhöz és billentyűkombinációkhoz kössék. Tették ezt annál is inkább, mert majd húsz évvel ezelőtt még nem voltak ismertek a grafikus képernyők és a mutatóeszközök. A végeredmény az lett, hogy mára igen nagy számú Emacs-parancs és -változó áll a felhasználók rendelkezésére, akik így igen sok feladatot tudnak egyszerűen és hatékonyan megoldani. Cserébe viszont a kezdőknek viszonylag hosszú ideig kell ismerkedniük az Emacs-környezettel, és nemcsak a számtalan parancs nevét kell megtanulniuk, de a hozzájuk tartozó billentyűkombinációkat is készségszinten el kell sajátítaniuk.

Gondoljuk csak végig, hogy másképp van-e ez a többi, felhasználóbarátnak tartott alkalmazásban! Vajon a grafikus felületek kitalálói jobb receptet kínálnak a fenti dilemma orvoslására? Hatásos megoldás lehet, ha csökkentjük a felkínált parancsok számát – áttekinthetőbbé és könnyebben megtanulhatóvá téve a programot. Az ilyen lebutított változatok kielégítők lehetnek az átlagfelhasz-

nálók, de nem a szakemberek számára. Megpróbálkozhatunk a „szolgáltatáselrejtés” alkalmazásával, amit én a Microsoft Word 2000-ben láttam először.
A szakembereknek szánt programcsomagok viszont valósággal kérkednek az elérhető szolgáltatások százaival. Vessünk például egy pillantást a ***Blender*** nevű 3D-modellezőre, amely egyszerre több tucat nyomógombot rak ki a képernyőre, és ezek a gombok az üzemmódtól függően állandóan változnak, mindig újabbak bukkannak elő!


6.

Hiába grafikusak ezek a felületek, igazából nem lehet őket felhasználóbarátnak nevezni, hiszen előtanulmányok nélkül a zöldfülűek semmit sem tudnak kezdeni velük. Az ilyen programokat azonban az irodai programcsomagokkal ellentétben nem akarják mindenkinek eladni, beleértve az olvasni tudó kisdedeket és a homályosodó szemű aggastyánokat. A szakembereknek fejlesztő programozók számára tehát nem az a kérdés, hogy megmutassák-e a programban rejlő lehetőségeket, hanem az, hogy miképpen tegyék meg. Általánosan elfogadott módszer, hogy a szolgáltatásokat ilyen-olyan szempontok szerint csoportosítják, majd menükre, gyorsbillentyűkre, ikonsorokra vagy nyomógombokra fűzik fel őket. A párbeszédablakos megjelenítés kényelmes és felhasználóbarát, ha ritkán használt beállításokról van szó, de igen hátráltatja a munkát, ha gyakorta ismétlődő parancsok előhívására használjuk. Gondoljunk bele például, hogy a kijelölt szövegrész kivágásakor mennyire időigényes mozgássor az F10 funkcióbillentyű megnyomása, majd a kivágás menütétel kiválasztása a NYÍL billentyűkkel, és végezetül az ENTER leütése. Gyorsabb, ha a kivágás ikont nyomjuk meg gépeléskor, de még ilyenkor is oda kell vinnünk a kezünket az oldalt lévő egérhez, ami miatt meg kell szakítanunk a szövegbevitelt. Gépeléskor tehát a gyorsbillentyűk használata adja a legjobb eredményt, hiszen például az Emacsban elég megnyomni a CTRL-W billentyűkombinációt a kijelölt szövegrész kivágására, és az adatbevitel máris folytatható anélkül, hogy a kezünket egy pillanatra is arrébb kellett volna mozdítanunk. Mivel az Emacs fejlesztői tisztában voltak ezzel az ergonómiai törvényszerűséggel, nem nagyon törték magukat, hogy más, kevésbé hatékony, de mások által felhasználóbarátnak mondott megoldásokat keressenek. Érdekes módon az XEmacs sem törekszik erre, pedig azzal a céllal hozták létre, hogy könnyebben kezelhető legyen. Elég, ha egy pillantást vetünk rá, és rögtön látjuk, hogy a fejlesztők megelégedtek néhány ikonnal (összesen 15-tel), amelyeknek a felét én személy szerint fölöslegesnek tartom a mindennapi munka szempontjából. Miért kell például az Info ikont kirakni a szemünk elé, amikor nem tesz egyebet, mint megjeleníti a Súgót? Ezzel szemben a Microsoft Word szövegszerkesztőben nem 15 ikont, hanem 15 ikonsort találhatunk!
Az XEmacsban inkább a menük burjánzanak. Mégsem állítanám, hogy az XEmacs fejlesztői lusták lettek volna, amikor kispórolták az ikonokat. Inkább azt feltételezem, hogy nem látták értelmét annak, hogy ikonokkal vagy nyomógombokkal zsúfolják tele az új Emacs-változatot, hiszen az a programozói réteg, amelyik az Emacs-környezetet használja, nem igényel ilyen változtatásokat. A fejlesztők tehát a programozás és a leíráskészítés szempontjából a leghatékonyabb megoldást választották, azaz a gyorsbillentyűkkel való parancshívást, még akkor is, ha ez a kezdők számára első pillantásra elrettentőnek tűnik. Nincs királyi út! Legfeljebb az Emacs.

Az Emacs és az Xemacs

A kétfajta Emacs között nincs nagy különbség, de hamar észrevehetjük, hogy nem teljesen egyforma a kettő. Az XEmacsban hiába kerestem a szöveges állományokat átalakítás nélkül beolvasó `find-file-literally` parancsot, csak az Emacsban találtam meg. Bizonyos billentyűkombinációk is másként működnek a két változatban, de összességében nem nagyok az eltérések. Tanulás közben célszerű mindkettőt kipróbálni, és végül annál maradni, amelyik jobban tetszik nekünk.
Manapság a legtöbb alkalmazást valamilyen grafikus felhasználói felületre írják, ezért felmerülhet a kérdés, hogy van-e értelme az X Window nélkül valamelyik terminálról indítani az Emacsot, hiszen mindegyik ablakkezelőben van terminálemulátor, és abban már futhat az Emacs. De mi tehet az a programozó, aki éppenséggel egy ablakkezelő megírásába fog bele? Neki valószínűleg a parancssorról kényelmes tesztelnie új programját, hiszen éppen fejlesztés alatt álló ablakkezelője feltehetően még nem igazán használható. Ilyenkor csak valamelyik parancssori szerkesztőt futtathatja.
Ha nincs egerünk, akkor a terminálon indított Emacs még akkor is új élményt fog jelenteni számunkra, ha már jártasak vagyunk a terminálemulátorban futtatott Emacs használatában. Megszoktuk már, hogy a legtöbb alkalmazásban az F10 nyitja le a menüt, és nincs ez másként az Emacsban sem. A parancssorra tévedt felhasználó azonban meglepetten tapasztalja, hogy hiába nyomkodja az F10 funkcióbillentyűt, semmi sem változik a menüsorban. Minél feszültebben figyel azonban a képernyő felső részére, annál kevesebb esélye lesz arra, hogy észrevegye, mi történik az alsó fertályban. Az Emacs ugyanis a képernyő alján, a mini átmeneti tárban (pontosabban a visszhangterületen) írja ki a menüket, és ott is kell választanunk a megjelenő menütételek a balra vagy jobbra nyilak közül, majd az ENTER megnyomásával.
Hasonló furcsaságokkal bármikor találkozhatunk az Emacsnál, de ha már megismertük, többé nem fognak meglepődni.

Utószó

Nyilvánvaló, hogy a vi vagy az Emacs más, mint amit a többi operációs rendszerekben megszoktunk, de az a tény, hogy furcsa és szokatlan, még nem lehet a minősítés alapja. A szövegben szereplő kódbetűs szavak nemcsak a programok nevére utalnak, hanem megegyeznek a héjba beírandó programindító parancsokkal, a nagybetűvel kezdődő szavak viszont csak a programok neveit jelölik.
A következő részben röviden ismertetem az Emacs használatát.

Szaló István
(ratiosoft@freemail.hu)
tanár, immár több mint másfél évtizede foglalkozik programozással, de csak a Java és a Linux megismerése után tudta meg, hogy mi is az igazi programozás. Több írása megjelent már a hazai számítástechnikai lapokban. Ha néha feláll számítógépe mellől, rendszerint művészettörténész feleségével és kisiskolás lányával „találja szemben” magát.