Distribucije GNU/Linux-a za vgrajene sisteme

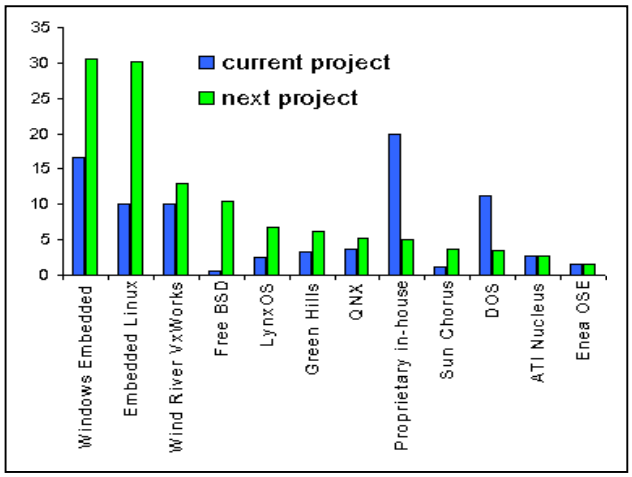
Rok Ostrovršnik, Martin Terbuc
Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Smetanova 17,Maribor
rok.ostrovrsnik@uni-mb.si,
martin.terbuc@uni-mb.si

Abstract: ***In this article we will try to show what embedded Linux is, where it is used and what is the current usage of Linux for embedded systems. We will also present good and bad sides of using Linux and impact of GPL on products developed with it. Article will also review two comercial embedded Linux toolkits.***

1. Uvod

Leta 1999, ko se je začelo govoriti o Linux-u in njegovi uporabi v vgrajenih sistemih, ni nihče pomislil, da lahko postane Linux resna grožnja komercialnim operacijskim sistemom. Komercialni operacijski sistemi za delo v realnem času so se začeli pojavljati proti koncu 1970 in danes jih je na voljo na trgu več deset. Največji igralci na tem področju so VxWorks, pSoS, Neculeus, Windows CE [Slika 1]. Glede na raziskavo podjetja EDC (Evans Data Corp), se delež Linuxa-a v vgrajenih sistemih nezadržno veča. Če je leta 1999 bil Linux le opcija, je danes povsem realno trditi, da lahko postane Linux vodilni OS na področju vgrajenih sistemov. V raziskavi [1], ki je bila opravljena v juliju 2002, se je 30,2% vprašanih razvijalcev vgrajenih sistemov opredelilo za Linux kot operacijski sistem za naslednji projekt. Za Windows CE se je opredelilo 16,6%, medtem ko bo Windows XP embedded uporabljalo v prihodnjih projektih 14,4% vprašanih. Windowsi imajo tako le neznatno prednost v primerjavi z Linux-om. Zanimivo, da celo VxWorks, ki je dolgo veljal za vodilnega na področju vgrajenih sistemov, rahlo zaostaja za Linux-om glede na trenutno uporabo po projektih. Pri prihodnjih projektih pa se bo razlika občutno povečala v korist Linux-a. Potrebno pa je omeniti, da raziskava prikazuje le uporabo operacijskih sistemov po projektih in ne prihodke od licenc in orodij za projekt.

Spekter uporabe Linuxa v različnih napravah je res širok. Najdemo ga v dlančnikih (Sharp Zaurus), mobilnih telefonih (nova serija Motorolinih telefonov), domači zabaviščnih centrih (NEC, SONY), požarnih zidovih, robotiki in še bi lahko naštevali. Veliko zanimivih naprav z Linux operacijskim sistemom si lahko pogledate na internetnem naslovu www.linuxdevices.com, ki je posvečen Linux-u za vgrajene sisteme.



Slika 1

2. Prednosti Linux-a

Postavlja se vprašanje, kaj je naredilo Linux tako popularnega v tako kratkem času za uporabo v vgrajenih sistemih. Kot prvo je vsekakor treba omeniti, da pri Linux-u ni potrebno plačevati licence za operacijski sistem,

niti za izdelke, na katerih teče Linux. Licence, ki jih morajo izdelovalci plačevati za posamezno napravo z komercialnim operacijskim sistemom, so lahko od 1$ pa do 150$. Pri velikem številu kosov so tako lahko ti stroški precej visoki. Danes v času, ko je najpomembneje zmanjševati stroške v tem negotovem gospodarskem ozračju, je takšna prednost lahko ključnega pomena. Čeprav znižamo stroške, s tem prav nič ne trpi kvaliteta izdelkov, saj velja Linux skoraj že pregovorno za izredno robusten in zanesljiv operacijski sistem.

Modularnost je naslednja odlika, ki nam omogoča, da spravimo delujoč sistem z jedrom, ki ima podporo za mrežo in nekaj osnovnimi orodji v vsega 500k pomnilnika.

Še ena prednost pri uporabi Linux-a je, da spletna skupnost izredno hitro vključi podporo za nove arhitekture, IP protokole in različne naprave. Kot zanimivost omenimo, da je za Linux na voljo več gonilnikov naprav kot za katerikoli komercialni operacijski sistem [2]. Uporabnikom Linux-a so tako vedno na voljo najnovejši gonilniki, aplikacije in knjižnice, ki jih spletna skupnost vedno znova posodablja in izboljšuje. Ker je prenesen na različne arhitekture (Xscale, ARM, MIPS, PowerPC, SuperH…), pri izbiri arhitekture nismo omejeni. Bil je celo prvi sistem, ki je deloval na novih Intelovih 64-bitnih procesorjih Itanium. Ponudniki komercialnih operacijskih sistemov mnogokrat ne morejo slediti tempu novih tehnologij in posledično razvoju novih naprav, zato podpora le tem zaostaja. Razvijalcem, ki hočejo vedno uporabiti zadnji krik tehnike v svojih izdelkih, to mnogokrat ni všeč. Linux s tem ponavadi nima problemov, ker se spletna skupnost izredno hitro odzove.

Vsekakor pa so tu tudi slabosti, ki se pojavijo z uporabo Linux-a.

3. Slabosti Linux-a

Kot sem že omenil, je spletna skupnost izredno aktivna pri dodajanju podpore za nove naprave, popravljanju hroščev in uvajanju sprememb na splošno. Rezultat je, da skoraj vsakih 6 mesecev dobimo novo stabilno verzijo jedra. In ker se Linux ne boji podreti kompatibilnosti za nazaj, lahko nastajajo problemi pri prenašanju starih aplikacij in gonilnikov na nova jedra. Temu se lahko preprosto izognemo tako, da ostanemo na eni verziji jedra dokler res ne potrebujemo izboljšav, ki jih ponuja nova verzija.

Linux sam po sebi ni bil nikoli mišljen kot operacijski sistem za delo v strogem realnem času, zato ga brez modifikacij lahko uporabljamo zgolj za mehki realni čas oz. za aplikacije, ki niso časovno kritične. Komercialni operacijski sistemi, kot so VxWorks, QNX, LynxOS, so bili že v osnovi mišljeni kot RTOS (Real Time Operating System). Če potrebujemo za našo napravo strogi realni čas, potem je potrebno izbrati kakšno modificirano verzijo Linux-a [4]. Zaradi te potrebe se je pojavilo več verzij Linux-a, ki omogočajo delo v strogem realnem času. Med temi sta najbolj opazna RTLinux in RTAI.

Tretji problem je GPL, ki niti ni tako strašen kot večina misli in kar bomo še podrobneje pogledali. Strah pred GPL-om je večinoma posledica propagande velikih podjetij, ki se bojijo prevlade Linux-a in modela odprte kode. Ta podjetja širijo t.i. FUD (Fear, Uncertainty, Doubt), kar je povsem razumljivo in v njihovem poslovnem interesu. Problem, ki ga imajo, je ta, da konkurence ne morejo preprosto kupiti.

Veliko se jih pritožuje zaradi pomanjkanja standardizacije in porazdeljenega razvoja Linux-a. Zato se je 50 večjih podjetij v maju 2000 odločilo, da ustanovijo konzorcij, ki bi zgradil referenčno platformo (ELCPS-Embedded Linux Consortium Platform Specification) za Linux za vgrajene sisteme. Oblikoval se je t.i. ELC (Embedded Linux Consortium), ki danes šteje več kot 125 podjetij, med njimi tudi IBM, Siemens, Sony, Samsung. Poleg paltforme je njihova naloga tudi promocija Linuxa in njegove uporabe.

Ta platforma bo omogočila, da bomo lahko prenašali aplikacije med različnimi distribucijami Linuxa in različnimi ciljnimi sistemi. Seveda v kolikor bodo te distribucije v skladu z ELCPS. To bo mogoče zaradi uporabe enotnega API-ja. V času pisanja tega članka se je pojavila specifikacija ELCPS 1.0.

4. GPL in odprta koda

Linus Torvalds je izdal Linux jedro, ki je pokrito z GPL licenco, kar na kratko pomeni, da lahko uporabljamo in distribuiramo njegovo kodo brez plačila licenc, vendar pod pogojem, da daste kupcem na voljo tudi izvorno kodo. Če izvorno kodo spreminjate in jo prilagajate svojim potrebam, pade vaše delo avtomatsko pod GPL licenco, ker izhaja iz izdelka, ki je bil pokrit z GPL licenco. Tukaj se ponavadi pojavi strah, da so vsi izdelki, ki temeljijo na Linux-u, avtomatsko pokriti z GPL licenco. Če bi bilo to res, bi to pomenilo, da bi vsi naši konkurenti imeli vedno na voljo izvorno kodo. Vendar to vsekakor ni povsem tako. Poglejmo si najprej, kakšno posledico ima GPL na gonilnike, ki jih zelo verjetno rabimo pri svojem izdelku [5].

Pri Linux-u lahko gonilnik naprave uporabimo na dva načina. Lahko ga prevedemo skupaj z jedrom in je torej statično vključen v jedro. Druga možnost je, da ga prevedemo posebej in ga dinamično vstavimo v delujoče jedro. Pri prvem načinu naš gonilnik avtomatsko pade pod GPL licenco in moramo izvorno kodo razkriti. Če pa gonilnik vstavimo v delujoče jedro dinamično, se izognemo GPL licenci in gonilnik ostane naša intelektualna lastnina.

Podobno je pri programski opremi. Če je program povsem naš in ne vsebuje delov programov, ki so pokriti z GPL licenco, potem lahko naš program distribuiramo brez GPL licence, razen če si tega ne želimo sami. Če naš program vsebuje dele programov, ki so pokriti z GPL, potem tudi naš program avtomatsko pade pod GPL. Ravno tako lahko naš program prizadene GPL, če ga statično ali dinamično povezujemo s knjižnicami, ki so pokrite z GPL. Ker bi to pomenilo, da je skoraj nemogoče napisati program brez GPL licence, je večina knjižnic pokrita z LGPL licenco (tudi *glibc*), ki nam omogoča razvoj programske opreme brez implikacije GPL licence. Linux in GPL torej ne pomenita, da bomo morali naš ″know-how″ posredovati drugim.

5. Distribucije Linux-a za vgrajene sisteme

Pri razvoju naprav, ki bodo uporabljale Linux, imamo na voljo tri možnosti. Lahko sami zgradimo svojo distribucijo s pripadajočim jedrom, korenskim datotečnim sistemom in vsemi potrebnimi orodji, za kar se na svetovnem spletu najde mnogo priročnikov in navodil, kako to storiti. Druga možnost je, da vzamemo eno od nekomercialnih distribucij (ETLinux, uClinux, ThinLinux, PeeWee Linux,…). Tretja možnost pa je, da se odločimo za katero od komercialnih distribucij. Pri prvih dveh možnostih smo prepuščeni lastnemu znanju in podpori, ki jo najdemo na spletu. Ta je precej dobra, vendar lahko zataji ravno takrat, ko je to najmanj primerno.

Če torej Linux-a ne poznamo dobro in bi radi izdelek hitro spravili na tržišče, je tretja možnost najprimernejša.

5.1 Komercialne distribucije

Na trgu so prisotni štirje glavni ponudniki Linux distribucij za vgrajene sisteme:

- LynuxWorks (BlueCat)
- Embedix (Lineo)
- MontaVista Software (Hard Hat)
- RedHat (RedHat embedded Linux)

Ti ponudniki ne prodajajo Linux-a ampak podporo zanj in pa orodja, ki nam olajšajo delo z njim. Vse distribucije opravijo v osnovi enako delo: pripravijo jedro, zgradijo korenski datotečni sistem in vključijo potrebne aplikacije. Kako to storijo in kakšno podporo ter dokumentacijo vsebujejo, je različno od ponudnika do ponudnika. Vsi ponujajo preizkusne verzije svojih orodij. Dve od teh distribucij smo si pogledali tudi na FERI v Mariboru, zato jih bomo tu podrobneje opisali.

Ti distribuciji smo izbrali, ker so bile najboljše ocenjene tudi na različnih spletnih straneh. Če vas zanimajo tudi ostale, lahko več o njih zveste na naslovu [3].

5.1.2 BlueCat 4.0

Zadnja verzija BlueCat-a je zgrajena okoli jedra 2.4. Za prevajanje jedra in programov uporablja gcc verzije 2.95.3. Razhroščevanje poteka s klasičnim GNU gdb oz. DDD, ki omogočata daljinsko in lokalno razhroščevanje. Grajenje jedra poteka s klasičnim Linux vmesnikom (*menuconfig*, *xconfig*). Korenski datotečni sistem pa zgradimo tako, da v tekstovno datoteko vpišemo katere direktorije rabimo in kje bi radi imeli posamezne programe. Ko skonfiguriramo jedro in opišemo korenski sistem, poženemo skripte *mkkernel* (ki nam zgradi jedro), *mkrootfs* (zgradi korenski datotečni sistem) in na koncu *mkboot,* ki nam prekopira jedro in korenski datotečni sistem na izbrani medij. Pri tem doda tudi zagonski nalagalnik, ki ob zagonu naloži jedro v pomnilnik in začne z izvajanjem. Pri zadnjem koraku imamo na voljo različne opcije od kje bi radi Linux poganjali. Lahko se odločimo, da se bo Linux pognal iz diskete, diska ali Flash naprave, ali pa se bo naložil sistem ob zagonu iz mreže. BlueCat razvojno okolje je možno uporabljati tako v Windows kakor v Linux okolju.

Razvojna okolja, ki jih ponuja, so VisualLynux, ki je vstavek za Microsoft Visual Studio in omogoča razvoj aplikacij ter razhroščevanje iz Windows okolja. Lahko se odločimo tudi za CodeWarrior (če delamo v Linux-u je to edina izbira). Močno orodje je tudi SpyKer, ki je grafični analizator in nam omogoča, da iz oddaljenega računalnika opazujemo, kaj se dogaja z našim ciljnim sistemom, vse to pa prikaže v grafični obliki. Za njegovo uporabo ni potrebno jedra posebej prevajati, ampak le vstavimo modul v delujoče jedro. Prednost BlueCat Linux-a je, da ima zelo močna orodja za razvoj ter da omogoča povsem transparentno delo tako v Windows kot Linux okolju. Potrebno je še omeniti njihovo dokumentacijo, ki je izredno podrobna in kvalitetna. Slabost je, da moramo korenski datotečni sistem zgraditi praktično na roko in da ni orodja, ki bi preverjalo odvisnosti med posameznimi paketi. Zadnji problem so omejili s tem, da so pripravili več kot 20 delujočih konfiguracij, ki so nam lahko osnova za naš sistem. Večini je potrebno dodati le še svojo aplikacijo. Cena kompleta z SpyKer orodjem in 6 mesečno podporo je 5999$ za eno delovno mesto. Za večje količine so možmi popusti. Dodatno pa je potrebno dokupiti BSP-je (board support package), katerih cena je okoli 1500$. Na izbiro imamo veliko število podprtih arhitektur (Xscale, ARM, MIPs, IA-32, PPC, X86, …).

Če potrebujemo strogi realni čas, je na voljo verzija z RTLinux-om.

5.1.3 Embedix

Drugi paket, ki smo ga preizkusili, je uporabniku precej bolj prijazen. Omogoča delo samo iz Linux okolja, kar lahko pomeni problem za tiste, ki ne nameravajo zamenjati operacijskega sistema. Jedro, ki je uporabljeno, je 2.4, za razhroščevanje pa se prav tako uporablja gdb oz. DDD. Ves proces gradnje jedra, izdelave korenskega datotečnega sistema in prenosa na izbrani medij, poteka v grafičnem okolju. Za razvoj aplikacij je na voljo le Code Warrior. Prav tako podpira različne medije (disketa, disk, flash, disk on chip …) in različne ciljne sisteme (ARM, PPC, Xscale, X86,…).

Target wizard, ki nam omogoča grajenje jedra in korenskega sistema, med drugim tudi preverja odvisnosti med posameznimi paketi in odvisnosti paketov z izbirami v jedru. Tako se nam ne more zgoditi, da bi zgradili sistem brez podpore za mrežo s programi, ki jo potrebujejo. To se nam pri BlueCat-u lahko zgodi kar hitro. Ne omogoča pa nam grafično okolje takšne optimizacije glede velikosti kot neposreden pristop pri BlueCat-u. Grafični analizator procesov (GRPA), prikaže le procese, ki tečejo na oddaljenem ciljnem sistemu, ne pa tudi kako se izvajajo, kar nam omogoča SpyKer. Cena Embedixa je 5200$, vendar vsebuje le 30

dnevno podporo in omogoča delo le v Linux okolju. Podporo je seveda možno dokupiti. Cene BSP-jev so prav tako okoli 1500$.

Če potrebujemo strogi realni čas nam je na voljo RTAI.

6. Izkušnje

Glede na izkušnje ob preizkušanju lahko rečemo, da se da z obema paketoma precej udobno delati. Pri BlueCat-u sicer moti pretežno tekstovno delo, vendar, ko se ga privadiš, to ni več problem, poleg tega pa jedra in korenskega sistema ne gradimo ravno pogosto. Dokumentacija, ki je priložena, je podrobna ter kvalitetna in predvsem omogoči hitro delo. Možnost, da celoten proces opravljamo iz Windows okolja in znanega razvojnega okolja (Visual Studio), lahko le pomaga pritegniti več razvijalcev, ki so do sedaj oklevali zaradi bojazni, da se bodo morali priučiti novega operacijskega sistema.

Delo z Embedixom je sicer prijaznejše, vendar je ostal občutek, da ne omogoča takšne optimizacije kot BlueCat in omogoča le delo v Linux okolju, kar tudi ni tako slabo. Vsekakor mora vsakdo, ki bi se odločil za komercialno distribucijo Linux-a, tega prej preizkusiti in se sam odločiti, kaj mu ustreza.

7. Zaključek

Linux je bil ustvarjen pred več kot 10 leti. Od takrat ga je ogromno število programerjev izboljševalo, naredilo še bolj učinkovitega, stabilnega in robustnega. Če smo ga pred leti opazili le kot odlično strežniško platformo, se lahko prav kmalu zgodi, da bo postal prva izbira za razvijalce vgrajenih sistemov. K temu lahko veliko pripomore ELC s svojo referenčno platformo in model odprte kode, ki se je do sedaj izkazal za uspešnega. Razvijalci, ki uporabljajo Linux, lahko že danes uživajo POSIX kompatibilnost, zanesljivost in robustnost, veliko bazo aplikacij z izvorno kodo in vse to brez licenčnin.

Na trgu imamo na voljo veliko število raznih distribucij, tako komercialnih kakor tudi nekomercialnih. Vedno pa se lahko odločimo, da bomo ustvarili svojo Linux distribucijo ali celo novo verzijo Linuxa. Pri tem je omejitev lahko le čas in domišljija, zaprta koda vsekakor ni.

8. Literatura

[1] http://www.linuxdevices.com/articles/AT7342059167.html

[2] http://www-106.ibm.com/developerworks/linux/library/l-embl.html

[3] http://www.linuxdevices.com/articles/AT8402180338.html

[4] Rok Ostrovršnik,*Uporaba RTLinux-a v sistemih avtomatizacije,* diplomsko delo , Maribor 2002

[5] Craig Hollabaugh, *Embedded Linux: Hardware, Software, and Interfacing,* Addison-Wesley 2002