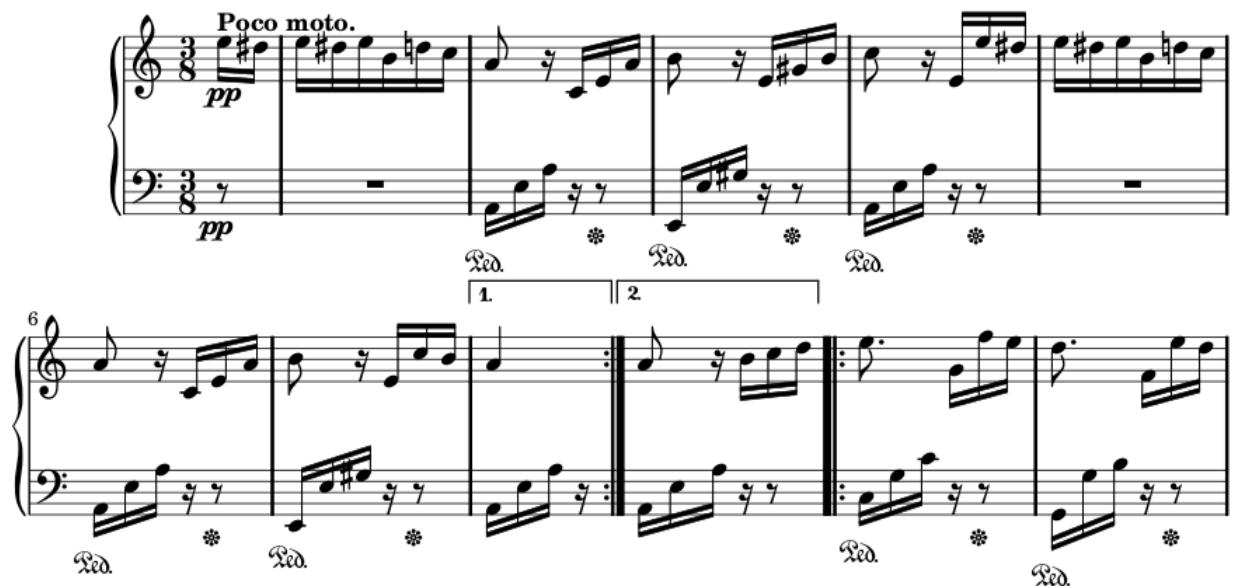


The OS Book

Sistemi Operativ hap pas hapi

Nicola Corriero - Eustrat Zhupa

10 Tetor 2011



Software is like sex: it's better when it's free

Copyright ©2011 N. Corriero & E. Zhupa

Të gjitha të drejtat e rezervuara.

ISBN 99921-58-10-7

Aulona Press

Indeksi

1 Hyrje	1
1.1 Dy fjalë mbi projektin	2
1.2 MVux	4
1.2.1 Portabiliteti	4
1.2.2 Përmasat	5
1.2.3 Qëllimi	5
1.2.4 Personalizimi	6
1.3 Përdorime të mundshme	6
2 Shënime mbi Sistemet Operative	9
2.1 Sistemet operative	10
2.2 Proceset	11
2.3 Kernel	15
2.4 Filesystem manager	17
2.5 Virtual memory manager	18
2.6 Skedulatori	19
2.7 Spooler	20
2.8 Shell	20
3 Linux	23
3.1 Linux: një paraqitje e shkurtër	24
3.1.1 Sistemi Operativ Linux	24
3.1.2 Multiuser dhe multitasking	27
3.1.3 Console dhe terminali	28
3.1.4 Të drejtat dhe atributet	30

3.1.5	Filesystem	31
3.2	Komandat themelore	35
3.2.1	Login dhe logout	36
3.2.2	Navigimi në filesystem	38
3.2.3	File dhe direktori	41
3.2.4	Shfaqja dhe printimi i files	44
3.2.5	Menaxhimi i proceseve	48
3.2.6	Ndryshoret e ambientit dhe alias për komandat	53
3.2.7	Të tjera komanda të dobishme	55
3.2.8	Faqet e manualit	62
3.3	Përbledhje e komandave kryesore	64
3.4	Shkarkimi	72
3.5	Si?	73
3.6	Kernel	73
3.7	Hard Disk	73
3.8	Grub	74
3.9	Bash	75
3.10	How To	76
3.11	Whys	77
3.12	Kernel	77
3.12.1	Konfigurimi	83
3.12.2	Përbërësit	84
3.13	Hard Disk	88
3.13.1	Probleme	91
3.14	BootLoader - Grub	92
3.15	Bash	95
3.16	Test	97

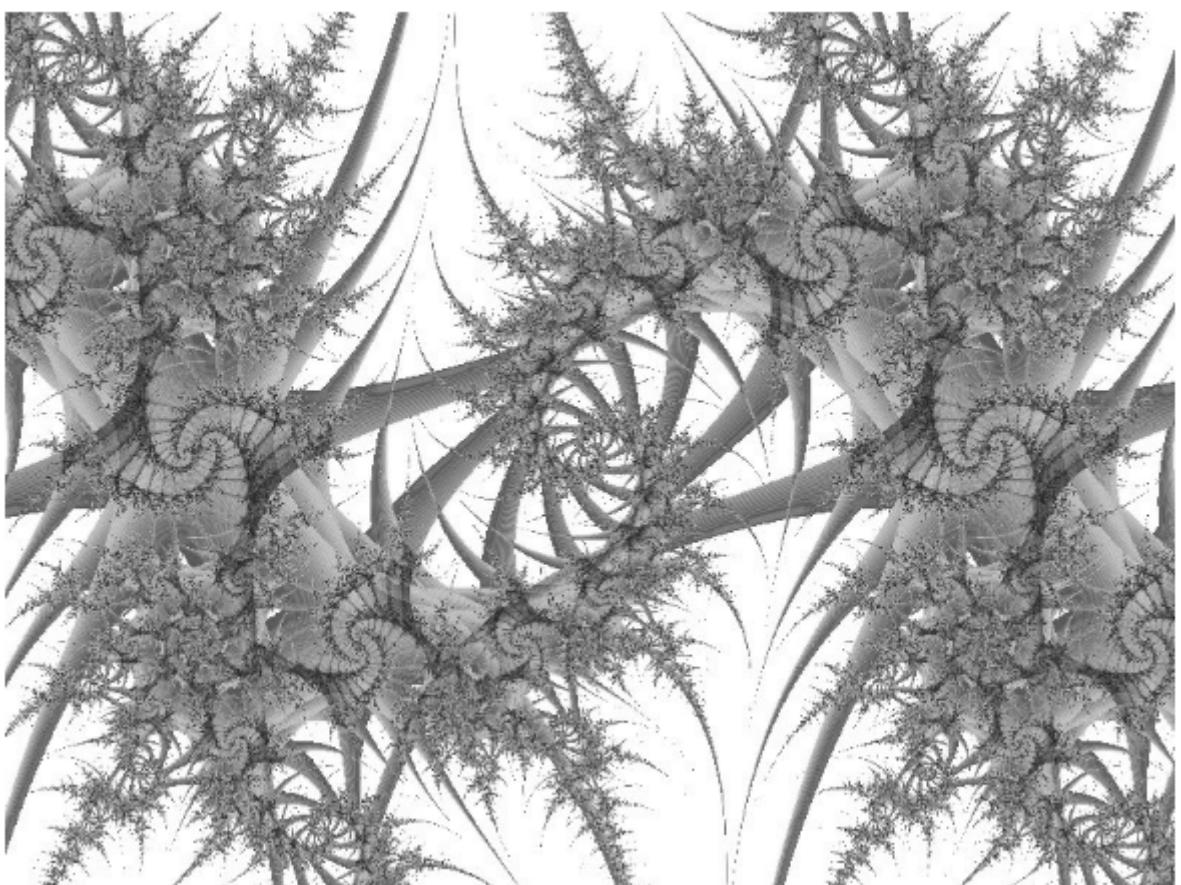
3.16.1 Konfigurimi	99
3.17 FAQ	100
3.18 Kernel	100
3.19 Hard disk	102
3.20 Grub	102
3.21 Bash	103
3.22 Coreutils	104
3.22.1 Shkarkimi	104
3.22.2 Si?	104
3.23 e2fsprogs	105
3.24 Util-Linux	105
3.25 Startup	108
3.26 FAQ - Boot	113
4 Ambienti Grafik	115
4.1 Server X	116
4.1.1 Pse?	116
4.2 Window Manager	118
4.2.1 Pse?	118
4.2.2 How to	118
5 Paketat	121
5.1 Si?	122
5.1.1 DPKG	122
5.1.2 Gcc	123
5.2 Pse?	125
5.2.1 Dpkg	125
5.2.2 Gcc	126

5.3	Programe të domosdoshme	127
5.4	Praktikë	128
5.4.1	Probleme	128
5.4.2	Disa vlerësime	130
5.5	Test	131
6	Rrjeti	133
6.1	Si?	134
6.1.1	Kompilimi i moduleve	134
6.1.2	Net-Tools	134
6.2	Si?	135
6.3	Test	136
6.3.1	Webserver e mini-chat	136
6.3.2	Socket	137
6.3.3	Ambienti i punes	138
6.3.4	Hyrja në rrjet nga pendrive	140
6.3.5	Realizimi i web server	142
6.3.6	Chat	154
7	Projekte të realizuara	163
7.1	Initrd	164
7.1.1	Konfigurimi Buildroot	164
7.1.2	Montimi i file imazh	165
7.1.3	Gjenerimi i nyjeve	166
7.1.4	Futja e skriptit dhe rregullat e udev	166
7.1.5	Modalitetet e ngarkimit të initrd	169
7.1.6	Problemet e hasura	170
7.2	Nfs	172

7.2.1	Konfigurim i Kernel	172
7.2.2	Networking	173
7.2.3	Klienti diskless	174
7.2.4	Serveri NFS	176
7.2.5	Skripti për autoboot	179
7.2.6	NFS dhe ndarja e pajisjeve	180
7.2.7	Nderfaqesi wifi	181
7.2.8	Perdorimi i initramfs	184
7.3	Busybox	187
7.3.1	Applet	188
7.3.2	Si të shtojmë një applet në BusyBox	189
7.3.3	Shtimi i Chmusic në BusyBox	189
7.3.4	Problemet e hasura dhe zgjidhjet	196
7.4	Instaluesi	197
7.4.1	Objektivi	197
7.4.2	Installer	198
7.4.3	Setup i skriptit	199
7.5	Rekuperim të dhenash nga sisteme operative që nuk startojnë	249
7.5.1	Objektivi	249
7.5.2	Krijimi i filesystem	251
7.5.3	Rikompilimi i kernel	257
7.5.4	Ntfs-3g dhe Fuse	261
7.5.5	File manager	264
7.5.6	Skripte	266
7.6	Motion Detection Linux	270
7.6.1	Ideja	270
7.6.2	Konfigurimi i Kernel	270

7.6.3	Problem me Inittab	271
7.6.4	Konfigurimi i rrjetit	273
7.6.5	Konfigurimi i webcam	274
7.6.6	Motion	274
7.6.7	Server ssh	276
7.6.8	Network Filesystem	277
7.6.9	Arkivimi i pamjeve	279
	Bibliografia	281

1 Hyrje



1.1 Dy fjalë mbi projektin

Ideja e krijimit të një *distribucioni* lindi si një nevojë për të plotësuar një laborator informatike me kompjutera të rinj. Në mënyrë që kompjuterat e vjetër të vazhdonin të përdoreshin, menduam të ndërtonim një distribucion në përshtatje me kushtet teknike dhe mundësitë hapësirë/shpejtësi që ofronin kompjuterat e vjetër. Qëllimi ishte që distribucioni të mund të mbahej në një hard disk të jashtëm (*pendrive*) me kapacitet të vogël. Kjo distro përmban të gjitha instrumentet e nevojshme për një student tipik të Shkencave Kompjuterike si: kompilatorë për disa gjuhë programimi, akses i kontrolluar në internet, një ambient bash për programimin shell dhe llogari përdorimi të personalizuara. Vlera e shtuar e këtij distribucioni janë përmasat. Hapësira e memories që nevojitet për këtë distribucion është prej pak MB. Përmasat e reduktuara bëjnë të mundur portabilitetin e sistemit edhe jashtë ambientit universitar, duke u dhënë mundësi studentëve të përdorin sistemin e tyre operativ kudo ku ndodhen dhe në mënyrë të pavarur nga kompjuteri pritës.

Realizimi i plotë i projektit varet nga aftësia e lexuesit për të kryer aktivitetet e mëposhtme:

- përdorim i komandave bazë në shell
- përdorimi eficent i manualeve dhe informacioneve mbi komandat
- konfigurim dhe kompilim i kernelit
- kompilim i kodeve dhe përdorim i **make** dhe instrumentave që kanë lidhje me këtë proces

Punimi që paraqitet është realizuar në një kuadër *open source* në nivel sistemesh operative të bazuar në kernel monolitik.

Sistemi ynë zëvendëson sistemet me përmasa të mëdha, siç janë distribucionet Linux në qarkullim, me një distro fleksibël që ofron një infrastrukturë lehtësisht të zgjerueshme dhe të modifikueshme: MVux.

Pasi një pendrive, që përbën MVux, vendoset në një kompjuter dhe pasi bëjmë zgjedhjet përkatëse në BIOS, sistemi operativ i pendrive starton duke na ofruar të gjitha shërbimet e caktuara. Pasi mbarojmë punët tona mund të shkëpusim pendrive nga sistemi dhe aty nuk do të ngelet asnje shenjë e aktivitetit tonë.

Në vijim të këtij teksti gjeni detajimin e hapave të nevojshme për të kriuar këtë distribucion, që ka edhe një ndërfaqes të thjeshtë.

Udhëzim për të kuptuar më mirë organizimin e librit.

Fillimisht në tekstu gjeni një paraqitje të thjeshtuar të koncepteve bazë, që gjeni në çdo sistem operativ. Në vijim, për çdo veprim do shpjegohen motivet (Teori); më pas do ilistrohen disa probleme që mund të hasen gjatë rrugës (Probleme); në fund, do të renditen në mënyrë shumë të detajuar veprimet e nevojshme për krijimin, duke u nisur nga kodet, e distribucionit (Kompilim).

Në seksionin e Teorisë mund të gjeni në fillim të çdo paragrafi, referimet për veprimet që do kryejmë, në mënyrë që teksti të jetë i lexueshëm dhe i kuptueshëm në çdohap. Në përfundim do të jepen disa indikacione për zhvillime të mundshme të distribucionit MVux.

1.2 MVux

Distribucioni MVux, objekt i këtij projekti, karakterizohet nga disa koncepte:

- Portabiliteti
- Përmasat
- Qëllimi
- Personalizimi

1.2.1 Portabiliteti

Në qarkullim mund të gjeni një seri mini-distribucionesh me sisteme minimale. Zakanisht këto karakterizohen nga fakti se ekzekutohen nga cdrom ose usb, por në modalitet *live*. Për shembull, ato nuk jepin akses për modifikimin e filesystem. Alternativa e tyre janë distribucionet e mëdha si Debian, Mandriva, Suse, Red Hat, Slackware, etj. Nëse nga njëra anë këto garantojnë eficencë dhe thjeshtësi përdorimi,

nga ana tjetër kanë nevojë për një hapesirë të madhe si në hard disk për ruajtjen e programeve, ashtu edhe në memorie për ekzekutimin e tyre. Distribucioni MVux, që është objekt i këtij projekti, lindi pikërisht për të plotësuar këtë boshllék. Sistemi është projektuar që të instalohet në një pendrive me kapacitet minimal, edhe në 256 MB. Kjo i jep mundësinë kujdo të mbajë në xhep një sistem operativ. MVux është validuar në ambiente të ndryshme: universitare, shtëpiake, punë. Mundësia për të përdorur të njëjtin sistem, me të njëjtat të dhëna, gjithnjë me të njëjtat programe, dekretoi suksesin e këtij distribucioni.

1.2.2 Përmasat

Me një zgjedhje projektimi të menduar mirë, sistemi u implementua në një pendrive me kapacitet të vogël. Natyrisht hapësira e zënë nga sistemi është proporcionale me sasinë e programeve të instaluara. Një tjetër zgjedhje ishte particionimi i pendrive në dy particione, njeri me filesystem *ext3* ku do instalohet distribucioni dhe tjetri me filesystem *fat32* ku mund të ruhen të dhëna. Në këtë mënyrë mund të përdoren edhe të dhëna që e kanë origjinën nga sisteme operative të tjera (win, mac, etj.).

MVux mund të startohet edhe në një kompjuter pa hard disk. Kjo mundësi na lejon, për shembull, të punojmë në kompjutera të vjetër, në të cilët nuk mund të instalohen sisteme operative të zakonshme. Një tjetër përdorim mund të jetë për rekuperimin e të dhënavë të dëmtuara ose të palexueshme. Vini re se mund të montohen hard disqe pa qenë nevoja që të modifikohen.

1.2.3 Qëllimi

Distribucioni MVux i realizuar prej nesh lind si një sistem për studentët e një laboratori informatik universitar. Pergjithësisht, në një laborator që shërben për një numër të madh studentësh të kurseve të ndryshme, duhen një numër i madh

kompjuterash, një sistem për menaxhimin e llogarive personale dhe memorizim të dhënash. Megjithëse ky projekt filloi në kuadrin e një ambienti të kufizuar, ai hedh baza të mira për përshtatjen e sistemit në një mori aplikimesh dhe fushash.

1.2.4 Personalizimi

Si në të gjitha distribucionet Linux, edhe në MVux përdoruesi ka mundësinë të personalizojë distribucionin e tij ashtu siç mendon. Prania e kompilatorëve dhe librarive të gjuhës C jep mundësinë për një programim në nivel sistemi. Në fakt, janë përfshirë në MVux edhe një seri librarish që bëjnë të mundur programimin grafik në C nëpërmjet dritareve dhe butonave.

1.3 Përdorime të mundshme

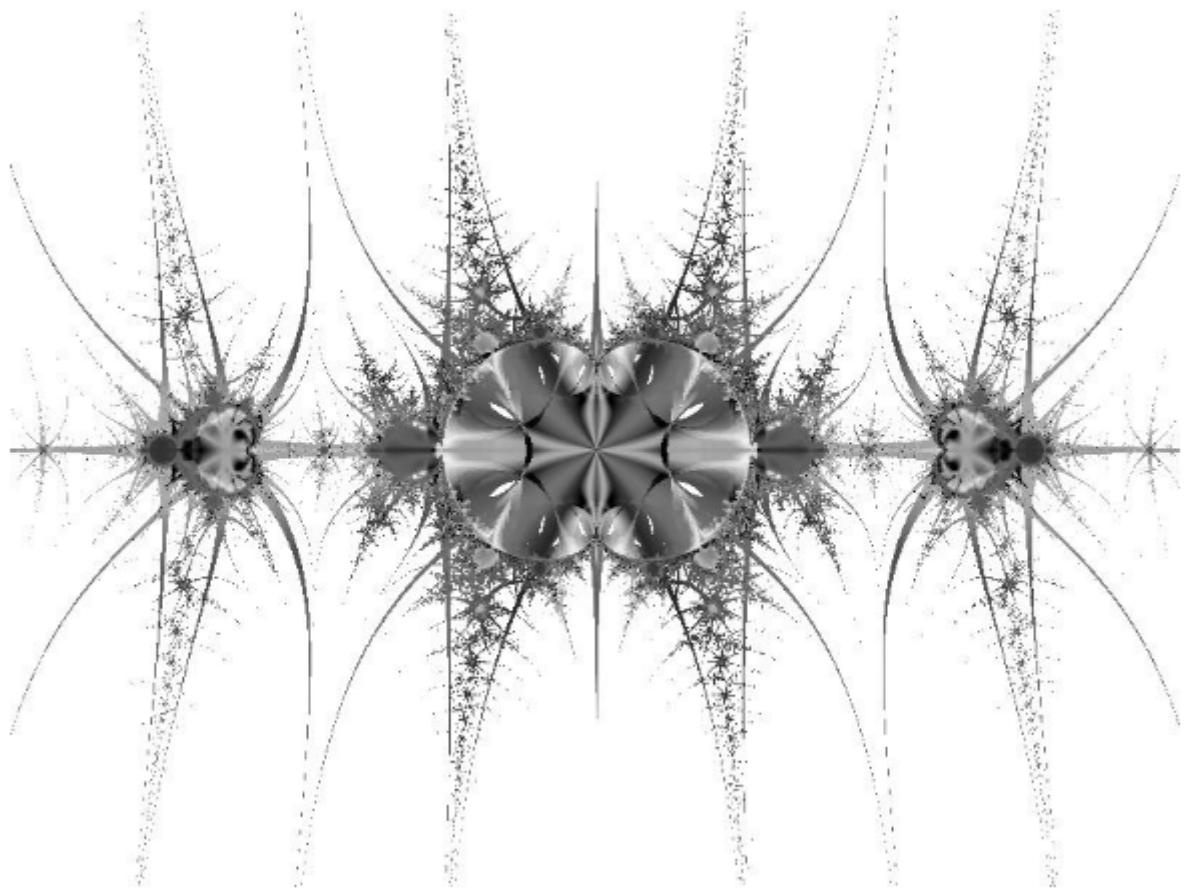
Ideja e MVux lindi me objektivin për të kupuar sistemet operative moderne dhe në veçanti për të kupuar se cfarë ndodh në kompjuter nga momenti i startimit deri në bashkëveprimin e parë me përdoruesin. Studimi i këtij fenomeni solli si rezultat krijimin e një distribucioni Linux të vogel, elegant dhe të shpejtë. Ky distribucion krijon mundësinë e përshtatjes për qëllime nga më të ndryshmet. Më poshtë paraqitet një listë aplikimesh praktike të distribucionit MVux:

- Bluetooth Marketing: një distribucion i thjeshtë për dërgimin e mesazheve publicitare, që shfrytëzon teknologjinë bluetooth. MVux u instalua në një kompjuter të vogël me dy antena bluetooth të lidhura për zbulimin e celularëve me bluetooth aktiv me qëllim dërgimin e mesazheve tekstuale të thjeshta.
- Rekuperim të dhënash nga sisteme operative: të gjithëve ju ka ndodhur të shihni sistemin tuaj operativ (jo open source) të bllokohet dhe të shfaqë mesazhe

gabimi të çuditshme. Të gjithëve ju ka ndodhur që të keni nevojë të reku-peroni të dhëna nga hard disk. MVux është përshtatur për rekuperimin e të dhënavët nga hard disk me sistem operativ jo funksional. Ideja bazë është që kompjuteri të startohet nga pendrive (me MVux të instaluar aty) dhe të rekuperohen të dhënat nga hard disk, që montohet nga shell.

- Sistem Diskless: MVux është përshtatur për krijimin e një laboratori me 10 kompjutera pa hard disk. Çdo kompjuter ka një usb me MVux të instaluar. Gjatë nisjes MVux, duke përdorur Network File System (NFS), mownton particionin nga një server i largët me aplikacionet që ekzekutohen në çdo kompjuter.
- MVux për *mobile*: MVux është përshtatur për të funksionuar në një arkitekturë ARM, e cila është tipike e pajisjeve celulare dhe smartphones. Në këtë mënyrë u bë e mundur të krijohej një sistem i thjeshtë portabël pa ndërfaqës grafik. Nëpërmjet studimit të MVux do bëhet e mundur të kuptohet dhe modifikohet startimi i Android, sistemi operativ i famshëm që bazohet në Linux. Në fakt, Android starton kur MVux përfundon ekzekutimin.

2 Shënime mbi Sistemet Operative



2.1 Sistemet operative

Në shkencën e kompjuterit **sistemi operativ**, shkurt SO ose OS (e dyta në anglisht *operating system*), është një grup programesh dhe struktura të dhënash që janë përgjegjëse për kontrollin dhe administrimin e përbërësve *hardware* të kompjuterit dhe për programet që ekzekutohen në kompjuter. Zakonisht sistemi operativ i krijon

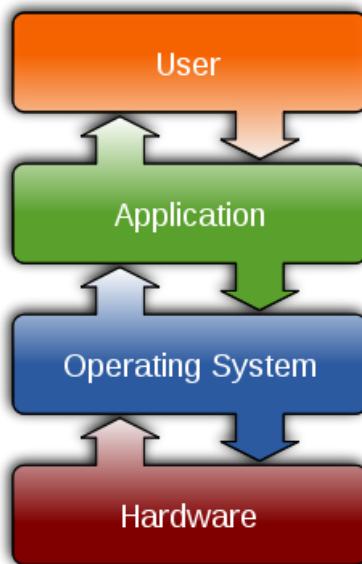


Figura 2.1. Paraqitje skematike

mundësinë përdoruesit të ketë në dispozicion një ambient grafik për të bashkëvepruar me pjesët hardware (disk, memorie, I/O, etj.) të sistemit. Këto mundësi varen nga tē drejtat që ka çdo përdorues në sistem. Detyra kryesore e sistemit operativ është t'i krijojë mundësi perdoruesit, qoftë ai njeri ose jo, të bashkëveprojë me kompjuterin. Në këtë kuadër mund tē kujtojmë disa prej sistemeve operative më tē përhapura në kompjuterat *desktop* si *Linux*, *Windows*, *Mac OS*, etj. Sisteme operative gjejmë edhe në pajisje të llojeve të tjera për përdorime të ndryshme. Në pajisjet celulare, që tē gjithë i njohim, mund tē gjejmë sisteme operative si *Windows Mobile*, *Symbian*,

etj.

Struktura e një sistemi operativ është e lidhur me një sërë përbërësish dhe kon-



Figura 2.2. Prodhues sistemesh operative

ceptesh si: *proces, kernel, filesystem, scheduler, spooler, memorie virtuale*. Në vazhdim paraqitet një panoramë e përgjithshme e këtyre elementeve.

2.2 Proceset

Me fjalën **proces** nënkuptohet një instancë e një programi gjatë ekzekutimit sekencial. Me fjalë të tjera, procesi është një aktivitet i kontrolluar nga një program dhe që zhvillohet në një procesor. Një program përbëhet nga kodi objekt i gjeneruar nga kompilimi i kodit fillestar shkruar në një gjuhë programimi, dhe normalisht ruhet në një ose më shumë arkiva. Programi është një entitet statik që nuk ndryshon gjatë ekzekutimit. Procesi është entiteti që përdoret nga sistemi operativ për të përfaqësuar një ekzekutim specifik të një programi. Pra është një entitet dinamik, që varet nga të dhënat që përpunohen dhe nga veprimet mbi to. Procesi karakterizohet, përveç kodit, nga tërësia e informacionit që përcakton gjendjen, si përbërja e memories së adresuar, *threads*, përshkruesit e arkivave dhe pajisjeve që përdoren

nga procesi.

Përdorimi i abstraksionit tek proceset është i nevojshëm për realizimin e **multiprogramimit**. Koncepti i procesit është i lidhur me konceptin e *thread* (ose ”fill ekzekutimi”), që nënkupton njësinë granulare në të cilën mund të ndahet procesi, dhe që mund të ekzekutohet paralelisht me *threads* të tjera. E thënë ndryshe, një *thread* është një pjesë e procesit që ekzekutohet në mënyrë konkurrente dhe të pavarur brenda të njëjtë proces.

Një proces ka të paktën një *thread* (veten e tij), por në disa raste procesi mund të ketë shumë *threads* që ekzekutohen paralelisht.

Një nga diferençat midis një *thread* dhe një procesi konsiston në mënyrën me të cilën ata ndajnë burimet. Ndërsa proceset janë zakonisht të pavarur nga njëri tjetri, përdorin zona të ndryshme në memorie dhe bashkëveprojnë vetëm nëpërmjet disa mekanizmash komunikimi që u jep sistemi, *threads* zakonisht ndajnë të njëjtat informacione mbi gjendjen, memorien dhe burimet e tjera të sistemit.

Diferencën tjetër thelbësore e gjejmë në mekanizmin e aktivizimit: krijimi i një procesi sjell gjithmonë ngarkesë për sistemin, sepse duhen caktuar të gjitha burimet e nevojshme për ekzekutimin (memoria, pajisjet periferike e kështu me radhë); kurse *thread* është pjesë e procesit, kështu që një aktivizim i tij bëhet në mënyrë të shpejtë dhe me kosto minimale për sistemin.

Më poshtë paraqesim disa përkufizime më formale:

- *procesi* është objekti i një sistemi operativ, të cilit i jepen të gjitha burimet e një sistemi për ekzekutimin e një programi, përvèç procesorit (CPU).
- *thread* është objekti i sistemit operativ ose i aplikacionit të cilit i jepet procesori për ekzekutim.

Në një sistem që nuk suporton *threads*, nëse duhet ekzekutuar njëkohësisht i njëjti program për disa herë, është e nevojshme të krijohen disa procese të bazuara

tek i njëjtë program. Kjo teknikë funksionon, por ka kosto të larta burimesh, sepse çdo proces duhet të ketë burimet e tij, por edhe sepse komunikimi midis proceseve e rëndon shumë sistemin dhe ngadalëson ekzekutimin.

Duke mbajtur shumë *threads* në të njëjtin proces, mund të merret i njëjti rezultat duke caktuar burimet vetëm një herë dhe duke shkëmbyer të dhënët midis *threads* nëpërmjet memories së procesit, që mund të përdoret nga të gjitha *threads*.

Një shembull aplikimi që përdor shumë *threads* është një browser Web, i cili përdor një *thread* të veçantë për të shkarkuar çdo figurë nga një faqe web që përbën shumë figura.

Një tjetër shembull është ai i proceseve server, që shpesh quhen shërbime ose *daemon*, që mund të përgjigjen njëkohësisht ndaj kërkesave që vijnë nga përdoruesit e ndryshëm.

Në një sistem multiprosesor, mund të arrihen përmirësime në performancë, në sajë të paralelizmit fizik të *threads*. Sidoqoftë, aplikacioni duhet të projektohet në mënyr të tillë që ngarkesa përpunuese të ndahet midis *threads*. Ky projektim është i vështirë dhe zakonisht shkakton gabime. Përveç kësaj, ekzekutimi i programit mund të jetë shumë i ngadalshëm; për këtë arsyë sot ka shumë pak *software* që shfrytëzon *threads* në sistemet multiprosesor. Në një sistem operativ *multitasking*, ka shumë *threads* që ekzekutohen njëkohsisht. Nga këta, maksimumi një numër i barabartë me numrin e procesorëve mund të ketë realisht kontrollin e procesorit. Pra proceset e ndryshme mund të përdorin procesorin për një periudhë kohe të kufizuar. Pra proceset mund të ndërpriten, të kalojnë në pauzë dhe të rifillojnë sipas algoritmave të skedulimit.

Gjendjet në të cilat mund të ndodhet një *thread* janë:

- në ekzekutim (*running*): *thread* ka kontrollin e një procesori
- gati (*ready*): *thread* është gati për ekzekutim dhe pret që skedulatori ta kalojë në ekzekutim

- pritje (*suspended*): *thread* ka ekzekutuar një *system call* dhe ka ndaluar duke pritur rezultatin

Me termin **ndryshim konteksti** (*context switch*) quajmë mekanizmin nëpërmjet të cilit një *thread* në ekzekutim ndalohet, sepse ka ekzekutuar një *system call* ose sepse skedulatori ka vendosur të ekzekutojë një tjeter *thread*), kështu që një tjeter *thread*, që ishte i ndaluar, vazhdon ekzekutimin.

Gjatë *bootstrap* të sistemit operativ janë në ekzekutim një ose disa procese të krijuara nga sistemi operativ. Gjatë startimit të sistemit, në bazë të konfigurimit, mund të krijohen shumë procese. Gjatë veprimtarisë normale, në bazë të kërkessave të përdoruesve, mund të krijohen procese të tjera dhe disa mund të përfundojnë ekzekutimin e tyre.

Në momentin që sistemi operativ fillon ekzekutimin e një programi, krijon një proces me një *thread* të vetëm. Gjatë ekzekutimit të këtij *thread*, që quhet *thread kryesor*, kodi i programit mund të krijojë *threads* të reja ose procese të tjera nëpërmjet *system calls*.

Krijimi i një procesi ndryshon nga një sistem operativ në tjetrin. Në Windows përdoret një *system call* i quajtur **CreateProcess**, me të cilin specifikohet emri i file që përmban programin e ekzekutueshëm; ky file ngarkohet në memorie dhe ekzekutohet. Në Unix, përdoret një *system call* i quajtur **fork** për krijimin e një procesi të ri fëmijë, identik me procesin që ka thirrur *system call*; përdoret në vijim thirrja **exec** në njërin prej tyre për të ngarkuar kodin e ekzekutueshëm të një programi të ri në procesin aktual dhe për ta ekzekutuar.

Nga ana tjetër, krijimi i një *thread*, është një procedurë më uniforme. Në fakt, edhe thirrja **CreateThread** e Windows edhe thirrja **thr_create** e Solaris (një variant i Unix) kërkojnë kalimin e adresës për të njëjtat routine, të përmasave të stack dhe parametra të tjerë. *System call* bën të mundur ekzekutimin e routine që specifikohet, në mënyrë konkurrente me kodin që ekzekutohet nga thirrësi.

Në sistemet Unix-like, kur një proces përfundon, sistemi operativ liron burimet e zëna, perveç PID dhe Process Control Block (PCB). PCB ngelet në tabelën e proceseve (process table) në mënyrë që procesi prind të lexojë exit status duke thirrur `wait()`. Pas kësaj thirrje edhe PID dhe PCB lirohen për t'u përdorur nga procese të tjera. Derisa të ndodhë kjo procesi është në një gjendje që quhet ”zombie”.

Kur një proces përfundon para procesit fëmijë, ky i fundit bëhet një proces i ashtuquajtur ”jetim” dhe në sistemet Unix-like, adoptohet automatikisht nga procesi special i sistemit i quajtur `init`.

2.3 Kernel

Kernel është një grup funksionesh themelore, të lidhura ngushtë midis tyre dhe me hardware, që ekzekutohen me privilegjin maksimal ose siç thuhet ndryshe në **modalitet kernel**. Kernel ofron funksionet bazë për të gjithë përbërësit e tjerë të sistemit operativ, që realizojnë funksionet e tyre duke u bazuar në shërbimet e ofruara nga kernel. Në bazë të llojit të sistemit operativ, kernel mund të përfshijë pjesë të tjerë (kernel klasik, monolitik ose modular) ose të ofrojë vetëm funksionet bazë dhe të delegojë funksione të tjera në përbërës të jashtëm (microkernel). Në variantin **monolitik** përcaktohet një ndërsaqe virtuale e nivelit të lartë në *hardware*, me një bashkësi instruksionesh primitive ose system calls për të implementuar shërbimet e sistemit operativ si administrimi i proceseve, multitasking dhe administrimi i memories, në modulet e ndryshme që ekzekutohen në **modalitet supervizor**. Edhe pse çdo modul që realizon këto shërbime është i ndarë nga pjesa tjetër, integrimi i kodit është shumë i ngjeshur dhe është vështirë të bëhet në mënyrë korrekte. Duke qenë se të gjitha modulet operojnë në të njëjtën hapësirë, një *bug* në një prej tyre mund të bllokojë gjithë sistemin. Sidoqoftë, nëse implementimi është i plotë dhe i sigurt, integrimi i ngushtë i përbërësve e bën një kernel monolitik shumë eficent. Dizavantazhi

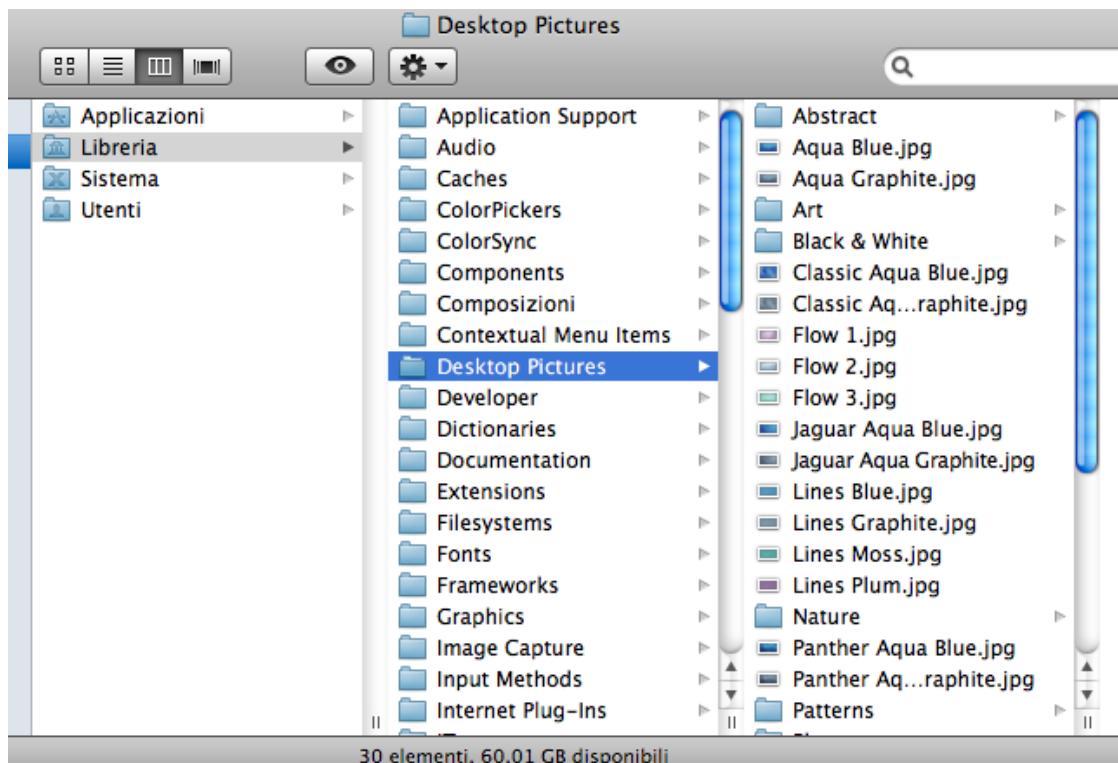
më i madh i kernelit monolitik është se çdo herë që shtohet një përbërës i ri hardware duhet të shtohet moduli përkatës dhe kjo kërkon rikompilimin e kernel. Një tjetër mundësi është kompilimi i kernel me të gjitha modulet që suportojnë hardware. Ky variant i rrit shumë përmasat e tij. Sidoqoftë kernel monolitike si Linux Kernel dhe FreeBSD mund të ngarkojnë modulet në fazën e ekzekutimit, me kusht që këto të jenë parashikuar në fazën e kompilimit, duke bërë të mundur zgjerimin e kernel sipas nevojave. Në këtë mënyrë përmasat e kernel janë të reduktuara në minimumin e nevojshëm. Tipologja me **microkernel** konsiston në përcaktimin e disa makinave virtuale shumë të thjeshta mbi *hardware*, me një set instruksionesh primitive për implementimin e shërbimeve minimale të sistemit operativ si administrimi i *threads*, hapësirave të adresimit ose i komunikimit midis proceseve. Objektivi kryesor është ndarja e implementimit të shërbimeve bazë nga strukturat e sistemit operativ. Për shembull, procesi i mbylljes (*locking*) të Input/Output mund të implementohet si modul server në nivel përdoruesi. Këto module në nivel përdoruesi, të përdorura për të ofruar shërbime të nivlit të lartë për sistemin, janë modulare dhe thjeshtojnë strukturën dhe projektimin e kernel. Nëse një shërbim server nuk punon, ai nuk blokon gjithë sistemin dhe mund të rifillojë punën pavarësisht nga pjesa tjetër e sistemit. Kernel monolitike shpesh preferohen në krahasim me microkernel për shkak të një kompleksiteti të reduktuar në kontrollin e kodeve të hapësirës së adresimit. Për shembull XNU, kerneli i Mac OS X, bazohet në një kernel Mach 3.0 dhe ka përbërës BSD në të njëjtën hapësirë adresimi në mënyrë që të zvogëlohen kohët e latencave, që janë tipike për microkernel. XNU është një kernel me performancë të mirë, sepse bazohet pjesërisht në një zgjidhje hibride dhe nuk mund të konsiderohet një microkernel. Në dokumentacionin zyrtar të Apple, XNU konsiderohet si një kernel monolitik modular. Duke filluar nga vitet '90 kernel monolitike konsiderohen si te vjetër. Projekti Linux si kernel monolitik dhe jo si microkernel është një

nga temat e diskutimit të famshëm midis Linus Torvalds (krijuesi i Linux) dhe Andrew Tanenbaum (profesor i njohur për sistemet operative, autor i Minix). Për këtë diskutim mund të gjeni informacion të gjerë në internet. Të dy palët në diskutim paraqesin motive të vlefshme dhe të arsyeshme. Kerneli monolitik lehtëson projektimin dhe zhvillohet më shpejt në krahasim me sistemet e bazuara në microkernel. Të dyja implementimet kanë arritur suksese gjatë historikut të tyre. Microkernel shpesh përdoren për sistemet embedded në aplikime *mission critical* në robotikë ose mjekësi, sepse përbërësit e sistemit janë të vendosura në zona të mbrojtura në memorien e sistemit. Kjo nuk është e mundur me kernel monolitik, as nëpërmjet përdorimit të moduleve.

2.4 Filesystem manager

Filesystem manager është përbërësi i sistemit operativ, që administron kërkosat për përdorimin e memories. Hyn në funksion sa herë që hapet një arkiv në disk dhe pervez transmetimit të të dhënave, regjistron edhe të drejtat e përdoruesve për aksesim të arkivave. Gjithashtu, filesystem manager realizon abstraksionin logjik të të dhënave të memorizuara në kompjuter.

Në shkencën e kompjuterit, **filesystem** është një mekanizëm që rregullon mënyrat se si memorizohen dhe organizohen arkivat në një pajisje arkivimi, në një disk, etj. Me një përkufizim më formal, ”filesystem është një bashkësi të dhëna abstrakte të nevojshme për memorizimin, organizimin hierarkik, manipulimin, aksesimin dhe leximin e të dhenave”. Në fakt, disa filesystems, si për shembull NFS, nuk bashkëveprojnë në mënyrë të drejtpërdrejtë me pajisjet e arkivimit. Filesystems mund të paraqiten grafikisht ose tekstualisht nëpërmjet nje browser ose në shell. Në paraqitjen grafike përgjithësisht përdoret metafora e arkivave që përbajne arkiva të tjerë dhe files.



Shembull i pemës së filesystem

2.5 Virtual memory manager

Memoria virtuale është një arkitekturë sistemi që realizon simulimin e një hapësire memorie qendrore më të madhe se memoria fizike reale; ky rezultat arrihet duke përdorur hapësira memorie dytësore në pajisje të tjera, zakonisht në njësi diskur. Memoria qendrore fizike është pjesa realisht e përdorshme në memorien virtuale: kjo teknikë realizohet duke u bazuar në principin e *lokalitetit të programeve*. Memoria dytësore e përdorur për këtë qëllim zakonisht quhet *swap* (në Posix) ose *hapësira swap* (folje në anglisht që ka kuptimin ”shkëmbej”), kurse në Windows quhet *paging file*. Veprimet e transferimit të faqeve nga hapësira swap në memorien fizike quhen *swapping*. *Administratori i memories virtuale* ka detyrën të caktojë memorien e kërkuar nga programet dhe nga sistemi operativ, të ruajë në memorien dytësore

zonat e memories që përkohësisht nuk përdoren nga programet, të garantoje se faqet e swap mund të risillen në memorie nëse kërkohen.

2.6 Skedulatori

Skedulatori (scheduler) është ajo pjesë e sistemit operativ, që shënon kohën e ekzekutimit të proceseve të ndryshme dhe garanton se çdo proces ekzekutohet sipas kohës së kërkuar. Normalisht skedulatori administron edhe gjendjet e proceseve dhe mund të ndërpresë ekzekutimin e tyre. Në sistemet operative *realtime* skedulatori garanton një *timeline*, pra një kohë maksimale përfundimin e çdo *task* në ekzekutim, prandaj është shumë më kompleks.

Skedulatori është pjesë shumë e rëndësishme në sistemet operative multitasking, pra që ekzekutojnë disa funksione në mënyrë konkurrente. Eshtë detyra e skedulatorit të kontrollojë avancimin e një procesi, duke ndërprerë një tjetër, dhe në këtë mënyrë realizon një ndryshim konteksti (context switch), siç u shpjegua më sipër. Në përgjithësi, kompjuterat me një procesor të vetëm mund të ekzekutojnë vetëm një program në një moment të caktuar. Për të bërë të mundur ekzekutimin e disa proceseve në paralel, lind nevoja e skedulatorit. Ekzistojnë algoritma të ndryshëm për skedulimin, që bëjnë të mundur ekzekutimin e shumë programeve në mënyrën me eficiente të mundshme. Për shembull, kerneli linux në versionin 2.4 ka një skedulator të rendit $O(n)$, ndërsa duke filluar nga versioni 2.6 algoritmi ka arritur rendin $O(1)$, e thënë ndryshe: përcakton në kohë konstante cili proces duhet ekzekutuar, pavarësisht nga numri i proceseve që janë në pritje.

2.7 Spooler

Përbërësi **spooler** është një program ose pajisje që përdor një sistem spool për të lehtësuar ose përshpejtar kalimin e të dhënave tek një pajisje me një shpejtësi të ndryshme ose me një linjë transmetimi jo konstante. Spooler shërben si ndërmjetës ose si ndërsaqe midis dy pajisjeve, merr të dhënrat nga burimi dhe i memorizon në buffer, duke liruar burimin. Në vijim i kalon të dhënrat konsumatorit me shpejtësinë e kërkuar.

Si shembull mund të konsiderojmë spooler të printimit, që merr të dhënrat nga programet dhe i printon në sekuencë, duke u krijuar mundësinë programeve të vazhdojnë ekzekutimin pa qenë nevoja të presin përfundimin e printimit.

2.8 Shell

Zakonisht sistemet operative ofrojnë një ndërsaqe (shell) për përdoruesin, që krijon mundësinë e bashkëveprimit të përdoruesit me kompjuterin.

Ndërsaqa grafike për përdoruesin, shkurt GUI (graphical user interface), është një praktikë zhvillimi që ka si qëllim të lehtësojë komunikimin e përdoruesit me kompjuterin.

Në një sistem operativ, **shell** (ose *terminal*) është një program që u jep mundësi përdoruesve të komunikojne me sistemin dhe të ekzekutojnë programe të tjera. Nëpërmjet shell mund t'i jepen komanda kompjuterit për kryerjen e veprimeve të ndryshme. Këto veprime realizohen kryesisht nëpërmjet një klase specifke programesh që quhen skripte shell (*shell scripts*)

Ka disa lloje shell, që i ndajmë kryesisht në *tekstuale* dhe grafike:

- kur përmendim fjalen ”shell” ose ”terminal”, zakonisht nënkuptojmë versionin e tyre ku përdoruesi bashkëvepron me kompjuterin nëpërmjet një ambienti

tekstual, ku e vetmja mundësi është të jepen komanda tekstuale;

- në rastin e shell **grafike** zakonisht gjejmë të ashtuquajturit *desktop environment*, që i japin mundësi përdoruesit të ketë në dispozicion një ambient grafik me të cilin mund të administroje arkiva, programe, etj.

3 Linux



3.1 Linux: një paraqitje e shkurtër

Në këtë kapitull paraqitet një panoramë e përgjithshme mbi strukturen teorike të një sistemi Linux. Duke ekzagjeruar disi mund të themi se paraqitet shkurtimish edhe ”filozofia e një sistemi Linux”.

Me siguri Linux është më i fuqishëm, më i qëndrueshëm dhe më eficent se Windows. Gjithashtu, Linux është më elastik dhe ndonjëherë më argëtues, si rrjedhim edhe më i vështirë për t'u mësuar; sidoqoftë, të punosh me këtë instrument të fuqishëm të jep kënaqësi të madhe: pasi të keni mësuar si të operoni në një sistem Linux, do jetë shumë e vështirë të pranoni kufizimet e shumta të sistemeve të tjera operative.

3.1.1 Sistemi Operativ Linux

Në vija të përgjithshme mund të themi se sistemi operativ është një *software*, i cili ngarkohët në memorie dhe ekzekutohet në momentin e ndezjes dhe nisjes së kompjuterit (*bootstrap*). Pasi hyn në funksion, sistemi bën të mundur ekzekutimin e aplikacioneve të tjera dhe shfrytëzimin prej tyre të burimeve *hardware* të kompjuterit. Në një farë mënyre, sistemi operativ i imponon përdoruesit mënyrën se si bashkëvepron me kompjuterin, duke përcaktuar çfarë mund të bëjë përdoruesi dhe çfarë nuk i lejohet të bëjë, rendin e veprimeve dhe kështu me radhë.

Disa nga këto aspekte nuk vihen re gjatë përdorimit të një kompjuteri personal me sisteme operative *single-user* si MS-DOS, versionet e para të Windows dhe versionet e vjetra të sistemit Apple: në këto ambiente mund të kryhen një numër i vogël veprimesh, por përdoruesi mund t'i kryejë të gjitha, pa asnjë kufizim.

Këto sisteme operative janë projektuar për të funksionuar në një kompjuter personal, pra në një kompjuter që përdoret nga një përdorues i vetëm. MS-DOS dhe versionet e para të Microsoft Windows, për shëmbull, ishin të konceptuara për

të ekzekutuar një program të vetëm (*task*) çdo herë¹: ndërkohë që përdorej një editor, nuk ishte e mundur të përdorej njëkohësisht një program komunikimi ose të ekzekutohej ndonjë program me *utility*.

Sot këto kufizime janë tejkaluar praktikisht nga të gjitha sistemet operative në qarkullim, por karakteristikat e multi-user dhe multitasking në kompjutera të përdorur nga një numër i madh përdoruesish (p.sh. kompjuteri personal) për shumë vjet kanë qenë ekskluzivë e sistemeve operative Linux. Linux lindi rrëth fillimit të viteve '70 njëkohësisht me futjen e *mini computer*, kompjuter më kompakt dhe fleksibël në krahasim me *mainframe* të prodhuar deri atëherë. Karakteristikat dalmesue të Linux, që përcaktuan edhe suksesin e madh të këtij sistemi, janë kryesisht veçoritë e tij si sistem operativ kompakt dhe modular, lehtësish i përshtatshëm me burimet hardware dhe me kërkeshat e ambientit ku instalohet. Me fillimet e Linux dhe mini computer fillon të përhapet edhe përdorimi i terminaleve alfanumerike, të cilat realizojnë një komunikim më të thjeshtë dhe eficent midis përdoruesit dhe sistemit (deri në atë kohë ishin ekskluzivisht kartat me vrima).

Gjithashtu, u përhap gjërisht koncepti i sistemit të hapur dhe përpunimi i shpërndarë (*distributed processing*): Linux është përgatitur për një komunikim të thjeshtë me sisteme të tjera dhe ndërlidhje me kompjutera të ndryshëm, për të punuar mbi burime jo të centralizuara në një sistem të madh, por të shpërndara në shumë kompjutera me fuqi mesatare; konfigurimi i sistemit dhe formatet e përdorura për files janë aspekte të njoitura e të dokumentuara (pra janë të "hapura") dhe cilido mund të ndërhyjë në to duke i përshtatur sipas nevojave².

Nëse do futeshim për një moment në detaje teknike, mund të theksojmë faktin se Linux është i bazuar në një bërthamë (kernel), i cili menaxhon komunikimin në nivel

¹Versionet e para të Windows nuk ishin një sistem operativ i vërtetë, por një ndërsaqe grafike për përdoruesin, e cila bazohej mbi një sistem operativ MS-DOS *mono-user* dhe *mono-tasking*

²Më shumë detaje mbi historinë e lindjes dhe zhvillimit të sistemit operativ Linux mund të shikoni faqet e Open Group në adresën http://www.unix.org/what_is_unix/history_timeline.html, ose në faqet e Bell Labs në adresën <http://www.bell-labs.com/history/unix/>

të ulët me kompjuterin, ekezekutimin e programeve dhe përdorimin dhe ndarjen ose mbrojtjen e burimeve të sistemit. Kjo është pjesa e sistemit operativ e lidhur më ngushtë me hardware të kompjuterit. Pjesa tjetër e sistemit përbëhet nga një seri modulesh shtesë, që kanë si qëllim menaxhimin e nënsistemeve të ndryshme, të cilat plotësojnë ambientin operativ (njojja e përdoruesve, shfaqja në video, input nga tastiera, ekzekutimi i komandave të përdoruesit, posta elektronike, printimi, etj.); këto module janë shpesh ”portabël”, pra mund të përshtaten pothuajse pa modifikime thelbësore për t’i përdorur në kompjutera plotësisht të ndryshëm, së bashku me kernel specifik. Ky fakt e bën Linux një sistem operativ të hapur dhe lehtësish të transportueshëm në hardware të ndryshëm. Eshtë pikërisht kjo largpamësi e projektuesve, e cila bën që sot të kemi shumë llojë sistemesh Linux, të përdorshëm në platforma të ndryshme. Nga ana tjetër, mundësia që ka çdokush për të marrë pjesë në projekt duke zhvilluar module të reja, bën që sot të mos ketë vetëm një Linux, por disa, dhe në disa raste mund të jenë pjesërisht të papajtueshëm midis tyre. Konceptet që eksposozohen në këtë guidë të shkurtër janë kaq të përgjithshme, sa nuk i nënshtrohen këtyre ndryshimeve të vogla. Mjafton të kemi parasysh se sot ekzistonë dy standarde referimi për zhvilluesit e sistemeve Linux: BSD, Berkeley System Distribution, dhe UNIX System V; ky i fundit (i njojur edhe si SVR4) ndoshta është më i përdorur nga BSD. Për secilën nga këto versione bazike të sistemit ezistojnë shumë implementime, disa shumë të njoitura dhe të përhapura, të tjera të njoitura dhe të përdorura vetëm nga një numër i vogël përdoruesish. Si shëmbull mund të themi se nga sistemet Linux të llojit BSD, më të njojur janë ato open source FreeBSD (<http://www.freebsd.org>), NetBSD (<http://www.netbsd.org>), Open BSD (<http://www.openbsd.org>), Darwin (baza Linux mbi të cilën ndërtohet sistemi operativ i Apple, Mac OS X; <http://www.opendarwin.org>, <http://www.gnu-darwin.org>); në variantin UNIX System V bazohen sistemi operativ (<http://www.linux.org/>), IBM AIX (<http://www.ibm.com/aix>), Sun Solaris (<http://www.sun.com/solaris>),

HP UX e Hewlett Packard (<http://www.hp.com/products1/unix/operating/>) dhe shumë të tjerë.

Shumë shpejt do vini re se Linux, në njëfarë mënyre, është një sistem serioz dhe elegant: çdo gjë që e bën Windows një sistem plot ngjyra dhe me efekt të madh viziv, në ambientin Linux shihet me dyshim. Këtu mbretëron sinteza, ku evitohet çdo gjë e panevojshme. Mesazhet e sistemit shfaqen vetëm kur është e domosdoshme: shpesh një program ekzekutohet pa dhënë asnë komunikim për përdoruesin; nëse ekzekutimi përfundon pa ndonjë mesazh, do të thotë se ekzekutimi i programit shkoi siç duhet. Në rast të kundërt, shfaqet një mesazh gabimi. Komandat e *shell* janë kompakte, shpesh kanë vetëm dy shkronja, por me një pafundësi opsjonesh, sepse përdoruesi duhet të jetë i lirë të modifikojë sipas dëshirës modalitetin operativ të software që përdor. Kjo e bën Linux një sistem operativ ekstremisht të përdorshëm dhe të gjithanshëm. Siç do shohim në vijim, një nga karakteristikat themelore të Linux është se ai paraqitet si një sistem operativ *multiuser*: shumë përdorues mund të përdorin sistemin njëkohësisht ose në kohë të ndryshme. Kjo bën që përdoruesit të detyrohen të zbatojnë disa rregulla në këtë sistem me shumë përdorues, ku burimet ndahen midis shumë personave. Fakti që shkëmbimi i mesazheve nga kompjuteri për te përdoruesi është i reduktuar në minimumin e nevojshëm, është një shenjë e idësë se çdo veprim i kryer në sistem, që ngadalëson punën e një përdoruesi tjeter, duhet evituar me kujdes. Në këtë mënyrë, lihen të lira sa më shumë burime që të jetë e mundur për ekzekutimin normal të programeve të tjera.

3.1.2 Multiuser dhe multitasking

Një sistem operativ **multitasking** mundëson ekzekutimin e shumë programeve (tasks) njëkohësisht: për shëmbull, nëse i kërkohët sistemit të ekzekutojë njëkohësisht dy procese, A dhe B, procesori ekzekuton për pak kohë procesin A, pastaj për pak kohë procesin B, më pas rikthehet te procesi A dhe kështu me radhë. Eshtë sistemi

operativ që kontrollon ndarjen e njëjtë të kohës nga ana e procesorit për të gjitha proceset aktive; është gjithmonë sistemi operativ që, nëse një proces blokohët për shkak të një gabimi, garanton paprekshmërinë dhe vazhdimin e punës për proceset e tjera. Një sistem me shumë përdorues, ose në anglisht **multiuser**, mund të përdoret njëkohësisht nga disa përdorues. Në Linux çdo përdoruesi të sistemit i jepet një llogari, që e identifikon në mënyrë univoke: kur fillon një sesion pune, përdoruesi duhet të ”futet” në sistem nëpërmjet një procedure *login*, gjatë të cilës përdoruesi duhet të identifikohët nga sistemi me një llogari (username) publike dhe me një fjalëkalim (password) të fshehtë. Pasi identifikohët nga sistemi, përdoruesi mund të fillojë punën duke ekzekutuar procese (aplikacione, programe), të cilat do të ekzekutohen në modalitet *multitasking* së bashku me proceset e përdoruesve të tjerë, që janë të lidhur me të njëtin kompjuter ose që kanë lënë në ekzekutim programe edhe pasi janë shkëputur nga terminali.

3.1.3 Console dhe terminali

Ndonjëherë mund të jetë e nevojshme të bëjmë dallimin e sistemit *hardware* që përdorim në kompjuterin me Linux ku do të punojmë. Në fakt, kur punojmë me kompjuterin tonë personal në ambientin Windows, jemi përdoruesit e vetëm të atij sistemi, dhe kompjuteri që po përdorim është ai që kemi përpara fizikisht³. Situata është shumë e ndryshme kur punojmë në një sistem Linux. **Console** është çifti tastierë/video i lidhur direkt me kompjuterin. Në ambientin Windows tastiera dhe ekranin i kompjuterit janë *console* e këtij kompjuteri. Duke qenë se në një kompjuter Linux mund të futen njëkohësisht disa përdorues, duhet të bëhet e mundur lidhja e disa tastiera/video në të njëtin kompjuter. Në fakt, në një sistem Linux zakonisht janë të lidhura disa terminale. **Terminali** përbëhet nga tastiera, video dhe nga

³Për thjeshtësi nuk konsiderojmë rastet kur kemi të bëjmë me sisteme Microsoft Windows Server.

një njësi përpunimi lokale, e cila ka si detyrë të vetme komunikimin midis terminalit dhe kompjuterit ku është i lidhur terminali. Në substancë terminali limitohet vetëm në shfaqjen e mesazheve të sistemit në ekran dhe në dërgimin e komandave të shtypura nga përdoruesi në tastierë. Në përgjithësi terminalet alfanumerike (ose ato që nuk mund të shfaqin pamje grafike, por vetëm tekstuale) janë të lidhur me kompjuterin nëpërmjet linjave seriale. Por shpesh si terminale përdoren kompjutera personalë, të pajisur me software të përshtatshëm për emulimin e terminalit. Në këtë rast fuqia e kompjuterit dhe e sistemit operativ të kompjuterit të përdorur si terminal nuk influencon, duke qenë se ai shërben vetëm për të transmetuar input nga tastiera dhe për të shfaqur rezultatet (output) në ekran. Përpunimi real bëhet në kompjuterin Linux ku është i lidhur terminali. Ekzistojnë terminale më të zhvilluar, të ashtuquajturit terminale grafikë, të cilët ofrojnë një ndërsaqe grafike (GUI) për të ekzekutuar veprimet e input/output dhe bëjnë të mundur marrjen e rezultateve edhe në formë grafike (imazhe, vizatime, grafikë). Terminalet grafikë mund të përbëhen nga pajisje hardware të projektuara qëllimisht (X Terminal), me njësi përpunimi grafik, ndonjëherë shumë të fuqishme, me ekrane me risolucion të lartë, tastierë dhe mouse; Terminalet X përgjithësisht janë të lidhur në rrjet TCP/IP me një ndërsaqs Ethernet me sistemin Linux; në këtë mënyrë mund të përdoren për futjen në modalitet grafik tek të gjithë serverat, që janë në rrjet dhe pranojnë lidhjen nga këto terminale grafikë. Një alternativë për Terminalet X janë software e quajtura X Window Server, që mund të ekzekutohen në një kompjuter personal (edhe në ambiente të ndryshme nga Linux, si Windows). Duke shfrytëzuar mundësitet e kompjuterit personal, bëjnë të mundur futjen në modalitet grafik me risolucion të lartë në aplikacionet grafike të pranishme në sistemin Linux. Në mbyllje, mund të përdoret si terminal grafik edhe console grafike e sistemit (nëse ka) ose ajo e kompjuterave të tjerë (workstation) Linux të lidhura në rrjet. Këto aspekte do trajtohen më në detaj.

3.1.4 Të drejtat dhe atributet

Përdoruesit e një sistemi Linux nuk janë të gjithë njësoj dhe mbi të gjitha nuk kanë të gjithë të njëjtat të drejta. Ky pohim mund të tingëllojë keq për ndonjë "demokrat" të sinqertë dhe ndoshta mund ta bëjë të mendojë se Linux nuk është një sistem operativ "liberal". Natyrisht këto mendime janë false: Linux garanton lirinë e veprimeve për të gjithë përdoruesit, duke bërë të mundur që asnje përdorues i paautorizuar të mund të cenojë *privacy* të përdoruesve të tjerë, ose të shkatërrojë e dëmtojë informacionet e të tjerëve. Në këtë mënyrë garantohet edhe një siguri e gjithë sistemit: asnje përdorues "normal" nuk mund të manipulojë sistemin duke dëmtuar funksionimin e tij normal; jo vetëm kaq: asnje përdorues "normal" nuk mund të bëjë gabime kaq të rënda në përdorimin e sistemit sa të dëmtojë përdoruesit e tjerë ose vetë sistemin. Më saktësisht: çfarë do të thotë e gjithë kjo? Kemi parë se çdo përdorues identifikohet në mënyrë univoke brenda sistemit nëpërmjet një llogarie (username). Përdoruesit e sistemit janë të shpërndarë në disa grupe; çdo përdorues bën pjesë të paktën në një grup. Ekziston edhe një përdorues i privilegjuar me username "root", që është administratori i sistemit, ndryshe *system manager*. Ky është figura më e rëndësishme në menaxhimin e një sistemi Linux. Eshtë përdoruesi që mund të modifikojë konfigurimin e gjithë sistemit dhe ka mundësinë të verifikojë çdo gjë brenda sistemit. Eshtë rasti për të thënë se është edhe i vetmi përdorues që mund të shkaktojë dëmtime serioze në një sistem Linux⁴. Si përkthehet kjo në praktike? Si fillim, çdo përdorues ka një hapësirë personale (directory) në filesystem të kompjuterit. Përdorues të ndryshëm kanë hapësira të ndryshme; këto hapësira (directory) janë private për përdoruesin dhe zakonisht kanë emrin e tij. Edhe çdo file tjetër në filesystem ka një "pronar" që, nëpërmjet komandave të përshtatshme,

⁴Ndoshta është mirë të saktësitet se, edhe pse files të përdoruesve të tjerë janë përgjithësisht të mbrojtur, është vështirë të evitojë mundësia që përdoruesi të dëmtojë files të tij. Duke punuar nga shell, nuk ekziston mundësia për të anulluar (undo) efektin e komandave ose të rekuperohen files nga koshi (trash)

përcakton modalitetet e përdorimit (atributet), të drejtat e aksesit, për vete dhe për përdoruesit e tjerë të sistemit. Eshtë e mundur, për shëmbull, të ndryshohen të drejtat për një file në mënyrë që ai të mund të modifikohët vetëm nga ”pronari”, të mund të lexohet nga përdoruesit e të njëjtë grup dhe të mos lexohet/shkruhet ose fshihet nga përdorues të tjerë jashtë grupit. E njëjtë gjë mund të bëhet edhe me *directory*, për të cilat mund të përcaktohen të drejtat e leximit, të shkrimit dhe aksesit, dhe për files të ekzekutueshme (programe), për të cilat mund të përcaktohet edhe e drejta e ekzekutimit. Përdoruesi ”root” nuk ka nga këto kufizime, por mund të aksesojë files dhe directory me të gjitha modalitetet e mundshme, pavarësisht nga atributet e mbrojtjes të përcaktuara nga përdoruesit. Eshtë mirë të kemi parasysh se, siç do shohim në vazhdim, çdo veprim i kryer në një kompjuter Linux kryhet në emër dhe për llogari të një përdoruesi të caktuar. Nuk ka ndonjë task ose program që ekzekutohet në mënyre anonime: çdo program ekzekutohet për llogari të një përdoruesi dhe ai menaxhon të gjitha të drejtat dhe kufizimet mbi të.

3.1.5 Filesystem

Shpesh kemi përdorur fjalën **filesystem**, por çfarë kuptimi ka realisht? Me këtë fjalë nënkuptojmë bashkësinë e pajisjeve të memorizimit, fizike ose virtuale, të lidhura me sistemin (teknikisht do thoshim: të montuara në sistem). Ata që kanë pasur rastin të përdorin një PC në ambient Windows, e dinë mirë se çdo njësi disku identifikohët nga një shkronjë e alfabetit: A është floppy disk, C është hard disk i parë, D është hard disk i dytë dhe kështu me radhë. Në ambientin Linux situata ndryshon në mënyrë radikale. Ekziston një njësi (disk) kryesore (*root*, ose rrënje e sistemit, që nuk duhet ngatërruar me përdoruesin me të njëjtin emër), në të cilën ”kaben” si subdirectory gjithë njesitë e tjera, qofshin ato njësi memorizimi të pranishme në kompjuter ose ”volume” të ndara në rrjet nga kompjutera të tjerë me protokolle të posaçme. Bashkësia e këtyre njësive të memorizimit (të quajtura ”volume”), të organizuara në një strukturë peme, përbëjnë atë që quhet *filesystem*. Përgjithësisht,

në çdo sistem Linux, gjejmë disa directory që kanë njëfarë rëndësie për sistemin dhe mbajnë përgjithësisht të njëjtin emër. Si shëmbull, konsiderojmë strukturën si pemë që shfaqet në skemën më poshtë, e cila paraqet një pjesë të një filesystem (në fakt disi i reduktuar).

```
\ ---+ bin
      |
      +--- dev
      |
      +--- etc
      |
      +--- home ---+ nicola
          |
          |
          |         +--- root
      +--- lib
      |
      +--- proc
      |
      +--- sbin
      |
      +--- tmp
      |
      +--- usr ---+ X11
          |
          |
          |         +--- bin
          |
          |         |
          |         +--- include
          |
          |         |
          |         +--- lib
          |
          |         |
```

```
|      +-+ local ---+ bin
|      |           |
|      |           +-+ lib
|      +-+ man
|
+-+ var ---+ log
|
|           +-+ mail
|
|           +-+ spool
```

Në vijim gjeni një shpjegim të shkurtër mbi përmbajtjen e directory që shihni në skemën më sipër:

/bin përmban pjesën më të madhe të *binary files* (të ekzekutueshme).

/dev është një directory shumë e rëndësishme, që mban gjithë *pointers* për *device drivers* të pajisjeve hardware të instaluarë në sistem. Këto janë disa nga shtesat e kernel që përmendëm më parë, të cilat bëjnë të mundur menaxhimin nga ana e sistemit të gjithë pajisjeve/njësive të lidhura në të. Për shëmbull, file `/dev/ttyS0` menaxhon input/output nga terminali i parë i lidhur me sistemin, ndërsa `/dev/console` menaxhon console të sistemit. `/dev/null` është njësia "bosh", që nevojitet në disa situata që do i shohim në vazhdim.

/etc përmban disa files të cilat nuk vendosen në directory të tjera; janë më tepër files konfigurimi të përgjithshëm të sistemit.

/home është directory e përdoruesit.

/lib përmban "libraritë" e sistemit, files që përmbajnë pjesë kodi të ekzekutueshëm dhe përdoren për kompilimin e aplikacioneve. Kjo zgjidhje redukton përmasat

e programeve, duke i organizuar në librari pjesët e kodit që janë të përbashkëta për shumë programe.

/proc është një directory e veçantë: files nuk janë të memorizuara në disk, por direkt në memorien e kompjuterit dhe përmbajnë referimet për proceset aktive në sistem dhe informacionet e nevojshme në lidhje me to.

/sbin përmban të gjithë files të ekzekutueshme të rezervuara për administratorin e sistemit.

/tmp është një directory e përkohshme. Shpesh aplikacionet duhet të shkruajnë të dhëna në një file të përkohshëm, që në përfundim të ekzekutimit do të fshihet; në këto raste përdoret directory **/tmp**, që është gjithmonë e pranishme në sistemet Linux.

/usr përmban disa directory shumë të rëndësishme për sistemin.

/usr/X11 përmban çdo gjë që ka lidhje me ndërsaqen grafike X Window.

/usr/bin të tjera files të ekzekutueshme (binary files).

/usr/include përmban files *include* për programet në C lidhur me libraritë e instaluar në sistem.

/usr/lib librari të tjera të sistemit.

/usr/local përmban files tipike të kompjuterit; zakonisht janë aplikacione të instaluar pas sistemit operativ.

/usr/man përmban faqet e manualeve për komandat e ndryshme.

/var përmban llojë të ndryshme files, përbajtja e të cilëve ndryshon shumë shpesh; këto files menaxhohen nga programe të tjera aktive në sistem.

/var/log përmban regjistrin ”historik” (logs) të eventeve që ndodhin në sistem dhe gjurmët e programeve aktive në sistem

/var/mail përmban files me *mailbox* të postës elektronike, me mesazhet e postës elektronike.

/var/spool përmban files në pritje për t'u perpunuar nga programe të tjera, si për shëmbull radha e printimit ose radha e mesazheve të postes elektronike në dalje.

3.2 Komandat themelore

Në këtë kapitull u japim një shikim të shpejtë komandave kryesore të shell në sistemin operativ Linux. Këto komanda na bëjnë të mundur ekzekutimin e programeve, zhvendosjen midis direktive të filesystem dhe menaxhimin e files dhe funksionet bazë të sistemit. Përtej funksionaliteteve të thjeshtuara që ofrojnë disa *desktop manager* në ambient grafik, mënyra më e mirë për të bashkëvepruar me një sistem Linux eshte nëpërmjet një shell. Ky është një program që interpreton dhe ekzekuton komandat e dhëna nga përdoruesi në bazë të një gjuhe, të cilën do e përshkruajmë në faqet në vijim. Shell përgjithësisht ekzekutohet në modalitet interaktiv, ose e përshkruar ndryshe: shell pret komandën nga ana e përdoruesit, pastaj e interpreton, e ekzekuton dhe rikthehet në gjendje pritje për komandën tjetër. Sidoqoftë, shell mund të përdoret edhe si një interpretues për ekzekutimin e një programi të plotë të shkruar në gjuhen e shell. Një program për shell do jetë një varg komandash dhe instrukcionesh, që teknikisht quhet skript (*shell script*). Shell është programi që zakonisht ekzekutohet automatikisht nga sistemi kur një përdorues futet në sistem. Kjo instancë specifike e shell quhet edhe *login shell*. Nëse do të bënim një paralelizëm me sistemin operativ Windows, mund të themi se shell loz të njëjtin rol si programi **command.com** ose **cmd.exe**, por është më i fuqishëm dhe i gjithanshëm. Skriptet

shell janë analoget e *batch files* (files *.bat) në ambientin Windows. Ka disa llojë shell për Linux në qarkullim: nga këto citojmë Bourne Shell (shkruar nga Stephen Bourne), që mund të thirret me komandën **sh**, bash ose Bourne Again Shell, një variant i zhvilluar i *sh*, Korn Shell *ksh*, C Shell *csh*, shumë e dashur mes programatorëve, sepse ka një sintaksë të ngjashme me atë të gjuhes C, *tcs*, zhvillim i C Shell, dhe të tjera. Në fakt, diferencat midis tyre janë të shumta, por këto dalin në pah vetëm nëse shkruajmë skripte, sepse për të tjerat janë shumë të ngjashme mes tyre. Eshtë e rëndësishme të thuhet se në ambient Linux shkronjat e mëdha dhe të vogla janë plotësisht të ndryshme: në Windows është njesoj nëse shkruajmë DIR ose **dir** dhe kjo vlen edhe për files. Në Linux nuk është njesoj: emrat e komandave janë *case sensitive* dhe duhen shkruar duke respektuar shkronjat e mëdha ose të vogla dhe e njëjtë gjë vlen edhe për emrat e files ose directory. Për më tepër, nuk ka kufizime për emrat e files: këta mund të jenë shumë të gjatë dhe nuk ekziston koncepti i zgjatimit (extension).

3.2.1 Login dhe logout

Siq u përmend me sipër, për të përdorur një kompjuter duhet të kryhet procedura e ”njohjes”, ose ndryshe **login**. Sistemi na kërkon **username** dhe **password**. Për te kuptuar me mirë shohim një shëmbull:

```
# login
mimi login: nicola
Password:
```

```
Last login: mar set 6 10.49.12 CEST 2011 su tty1
Linux mimi 2.6.38-2-686 #1 SMP Sun May 8 14:49:45 UTC 2011 i686
```

The programs included with the Debian GNU/Linux system are free software;

the exact distribution terms for each program are described in the individual files in `/usr/share/doc/*/copyright`.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

Nessuna email.

#

Fjalëkalimi nuk shfaqet në ekran, për të penguar zbulimin e tij nga persona që mund të jenë duke parë ekranin në ate moment. Kurse username është pjesa publike e llogarise se përdoruesit të sistemit. Ky informacion u shërben të tjereve për të komunikuar me ate përdorues. Mesazhi "Last login ..." na komunikon datën dhe orën e heres se fundit që jemi futur në sistem dhe emrin e terminalit nga i cili jemi lidhur, në këtë rast terminali `ttyS2`. Ky është një mesazh i dobishëm, sepse na lejon të kontrollojmë nëse është futur dikush tjetër në vendin tonë. Simboli "\$" që shfaqet para kurzorit është **prompt**, ose treguesi që shell është gati për të pranuar komanda nga përdoruesi. Ky që paraqitet në këto faqe është vetëm një shëmbull: prompt mund të ndryshojë nga një sistem në tjetrin dhe mund të konfigurohet nga përdoruesi. Zakonisht përdoruesi root ka një prompt të ndryshëm nga ai i përdoruesve te tjere. Kjo ndodh që përdoruesi të kuptojë se po përdor komanda që ekzekutohen me privilegje administratori. Nëse ai do të bënte gabime, mund të ketë pasoja katastrofike për integritetin e sistemit. Në shëmbuj do të përdorim simbolin "#" si prompt të përdoruesit root. Për të ndryshuar fjalëkalimin duhet përdorur komanda `passwd`. Sistemi do kërkojë të jepni fjalëkalimin e vjeter dhe, nëse ky është i saktë, do kërkojë dy herë (per siguri) fjalëkalimin e ri. Te dhëna si username dhe fjalëkalime të shifruara, bashkë me informacione baze mbi përdoruesit e sistemit, memorizohen në file `/etc/passwd`, që të gjithë përdoruesit mund të shohin. Në disa sisteme, për një siguri më të madhe, fjalëkalimet memorizohen në file `/etc/shadow`

(gjithnje të shifruara) dhe ky file mund të lexohet vetëm nga përdoruesi root. Pasi mbarojmë punen, para se të largohemi, duhet të shkëputemi nga sistemi, pra të kryejmë procedurën logout, që shërben pikërisht për t'i komunikuar sistemit se kemi mbaruar punen dhe mund të mbyllë sesionin dhe të kalojë në pritje të përdoruesit tjetër. Për t'u shkëputur mund të përdorim komandën `exit` ose komandën `logout`. Në disa shell kombinimi i tasteve Ctrl+D ka të njëjtin efekt. Pasi kryejmë logout, sistemi shfaq përsëri mesazhin e login në pritje për një lidhje tjetër në të njëjtin terminal.

```
# shutdown -h now
Broadcast message from root (pts/1) Tue Aug 10 07:38:16 2009...
The system is going down for system halt NOW !!
...
System halted.
```

3.2.2 Navigimi në filesystem

Pasi futemi në sistem, ndodhemi në direktori personal *home*. Për të verifikuar emrin e direktorisë ku ndodhemi, përdorim komandën `pwd` (print working directory).

```
$ pwd
/home/nicola
```

Simbol ”/” (slash), perveçse tregon direktorinë root në filesystem, përdoret edhe si ndarës midis direktive që bëjnë pjesë në të njëjtin *path*. Në sistemet Windows i njëji funksion kryhet nga simboli ”\” (backslash). Me siguri gjeja e parë që vjen në mendje është të shohim përbajtjen e direktorisë. Për këtë përdorim komandën `ls` (list), ekuivalente me komandën `dir` ne prompt të Windows. Për shëmbull mund të marrim një rezultat si me poshte:

```
$ ls
lettera      mail      pippo.c      progetto      tesi
libro       pippo      pippo.zip     src
```

Veme re se shfaqen nente emra, por nuk është e qarte nëse jane file të dhënash, nendirektori apo programe të ekzekutueshme. Përmendëm pak me parë opsonet e shumta që modifikojnë ekzekutimin e një komande ose programi. Në pergjithësi komanda shoqerohet nga një sekuence opsonesh me simbolin “-” perpara dhe nga parametra. Opsonet modifikojnë pjesërisht mënyrën e ekzekutimit të komandës. Për shëmbull mund të provojmë një variant të komandës `ls`:

```
$ ls -F /home/nicola
lettera      mail/      pippo.c      progetto@  tesi/
libro/      pippo*     pippo.zip     src/
```

Komanda është ”`ls`”, sekuenca ”`-F`” është opsiون dhe ”`/home/nicola`” është parameter i komandës. Opzioni ”`-F`” bën që prane disa emrash file të shfaqet një simbol që nuk bën pjesë në emer, por tregon llojin e file. Simboli ”/” tregon se është një direktori, ylli ”*” tregon se është një file i ekzekutueshëm, simboli ”@” tregon se ai file (ose direktori) nuk ndodhet fizikisht në direktorinë tonë, por është një link për një file (ose direktori) në një pozicion tjetër në filesystem. Do të kishim një tjetër version të komandës `ls` duke shtuar opzionin ”`-l`” (long). Duke qene se komanda pranon shume opsiون, le të shohim cili do jetë rezultati i komandës ”`ls -lF`”:

```
$ ls -lF
-rw-r--r--  1 nicola users   937 Apr 23 12:43 lettera
drwxr-xr-x  2 nicola users  1024 Apr 10 16:04 libro/
drwx----- 2 nicola users  1024 Feb 01 09:32 mail/
-rwxr-x---  1 nicola users 37513 Mar 10 11:55 pippo*
-rw-r--r--  1 nicola users 18722 Mar 10 11:30 pippo.c
```

```
-rw-r--r-- 1 nicola users 23946 Mar 10 12:03 pippo.zip
lrwxrwxr-- 1 nicola users      8 Apr 04 18:16 progetto -> /usr/proj
drwxrwx--- 2 nicola users   1024 Mar 10 08:47 src/
drwxr--r-- 2 nicola users   1024 Feb 12 1995 tesi/
```

Ilustrojmë shkurtimisht kuptimin e informacioneve që marrim nga komanda me sipër. Simboli i parë i çdo rreshti mund të jetë ”d” për të treguar se kemi të bejmë me një direktori, ”l” për të treguar një link, ose ”-” për të treguar një file normal. Shkronjat e tjera tregojnë në formë të përbledhur të drejtat mbi këtë file. Interpretimi behet duke i grupuar tre e nga tre. Tre të parat tregojnë të drejtat e ”pronarit” mbi ate file, tre të tjerat tregojnë të drejtat e përdoruesve të të njëjtës grup mbi ate file, tre të fundit tregojnë të drejtat e gjithë të tjereve mbi ate file. Shkronja ”x” tregon të drejtën e ekzekutimit, shkronja ”r” tregon të drejtën e leximit (read), shkronja ”w” tregon të drejtën e shkrimit/fshirjes (write) mbi ate file. Në vazhdim shfaqet username i ”pronarit” të file (p.sh. nicola) dhe emri i grupit ku bën pjesë përdoruesi (p.sh. users). Në të njëtin rresht shfaqet edhe përmasa e file, data dhe ora e e krijimit (ose modifikimit për herë të fundit) dhe në fund emri. Në shëmbullin me sipër, file ”pippo.zip” mund të lexohet dhe modifikohet nga nicola, ndërsa përdoruesit e sistemit vetëm e lexojnë. File është krijuar në datën 10 Mars 2010 në orën 12:03. File ”progetto” është një link për direktorinë /usr/proj dhe mund të përdoret në lexim, shkrim dhe ekzekutim vetëm nga pronari dhe nga përdoruesit e grupit *users*, por jo nga përdoruesit e tjere të sistemit që minden vetëm ta lexojnë. Nga opsonet e shumta të komandës ls po ilustrojmë edhe një tjetër, e cila na paraqet disa të reja në lidhje me emrin e file. Fillojmë me shëmbullin e zakonshëm duke parë rezultatin e komandës ”ls -a”:

```
$ ls -aF
./          .bashrc    lettera     pippo*      progetto@
../         .exrc      libro/      pippo.c     src/
```

```
.Xdefaults .newsrc mail/ pippo.zip tesi/
```

Me marreveshje në Linux (por edhe në Windows) me “.” (pike) shenojmë direktorinë ku ndodhemi aktualisht, ndërsa me “..” shenojmë direktorinë prind në pemen e filesystem. Ato files që kane vetëm “.” para emrit jane ”te fshehura” dhe shfaqen vetëm si efekt i opzionit ”-a” në komandën `ls`. Keto zakonisht jane files konfigurimi, që nuk përdoren. Për t’u zhvendosur mes direktorive në një filesystem përdoret komanda `cd` (change directory). Për shëmbull për të ”zbritur” në direktorinë `src` duhet të shkruajmë ”`cd src`”, ndërsa për t’u ”ngjitur” në direktorinë prind duhet të shkruajmë ”`cd ..`”. Persa i perket direktorisë sone *home*, asaj i referohemi edhe me simbolin ”~” (tilde), ndërsa për *home* directory të përdoruesit ”*marina*” mund të përdorim ”`~marina`”. Pra për t’u zhvendosur në direktorinë e përdoruesit ”*marina*” do të përdornim komandën ”`cd ~marina`”, e cila sigurisht është me e përbledhur sesa ”`cd /home/marina`”. Duke qene se shume shpesh kemi nevojë të zhvendosemi në direktorinë tonë, për të thjeshtuar veprimin kjo mund të behet vetëm duke shkruar ”`cd`” pa asnje opsjon ose parameter tjetër.

3.2.3 File dhe direktori

Sic pame në seksionin e meparshëm, çdo file ka disa atributë që rregullojnë aksesin nga ana e përdoruesve të sistemit. Për të modifikuar këto atributë përdoret komanda `chmod`, e cila përdoret me sintaksën ”`chmod attribute emri_file`”. Parametri *attribute* mund të ketë disa forma, por me e thjeshta është ajo numerike. Numri ndertohet me një rregull të thjeshte: numri 100 tregon të drejtën e ekzekutimit të file për pronarin, 10 për përdoruesit e grupit dhe 1 për gjithë të tjeret; numri 200 tregon të drejtën për shkrim të file nga ana e pronarit, 20 për përdoruesit e grupit dhe 2 për gjithë të tjeret; numri 400 tregon të drejtën e leximit të file për pronarin, 40 për përdoruesit e grupit dhe 4 për të tjeret. Duke bërë shumen e ketyre numrave, nxjerrim vleren numerike të atributit për ate file. Për shëmbull, atributi 700 na tregon se i vetmi që mund të

lexoje, të shkruaje dhe të ekzekutojë file është pronari ($100+200+400=700$), ndërsa vlera 644 na tregon se file mund të lexohet dhe modifikohët nga pronari, por vetëm mund të lexohet nga përdoruesit e tjere. Nëse duam të aplikojmë rastin e fundit për file ”*pippo.c*”, japim komandën:

```
$ chmod 644 pippo.c
```

Për të krijuar një direktori përdorim komandën **mkdir** (make directory). Për shëmbull me komandën

```
$ mkdir esempio
```

krijojmë një direktori brenda direktorisë ku ndodhemi.

Për të fshire një direktori bosh (qe nuk përmban asnjë file) përdoret komanda **rmdir** (remove directory).

```
$ rmdir esempio
```

Komanda ekzekutohet nga sistemi pa kërkuar ndonje konfirmim. Në rast se direktoria nuk është bosh, pra përmban files ose nendifektori, atehere komanda nuk ekzekutohet. Për të krijuar një kopje të një file përdoret komanda **cp** (copy). Për shëmbull nëse duam të kopjojmë ”*pippo.c*” në direktorinë *src* përdorim komandën:

```
$ cp pippo.c src
```

Një sintaksë të ngjashme ka edhe komanda **mv** (move) që shërben për të spostuar file nga një directory në një tjetër. Për shëmbull nëse duam të spostojmë të gjitha files qe mbarojnë me ”.c” nga *src* në *tesi* përdorim:

```
$ mv src/*.c tesi
```

Komanda **mv** përdoret edhe për të ndryshuar emrin e një file. Nëse duam të ndryshojmë emrin e ”*pippo.c*” në ”*pluto.c*” përdorim:

```
$ mv pippo.c pluto.c
```

Nëse duam të fshime një file përdorim komandën **rm** (remove). Duhet patur kujdes në përdorimin e kesaj komande, sepse sistemi nuk kërkon asnje konfirmim dhe e fshin direkt. Komanda e me poshtme fshin çdo file në direktori ”tesi”:

```
$ rm tesi/*
```

Duke përdorur opzionin ”-i” sistemi na kërkon konfirmim dhe pret një pergjigje ”y” (yes) ose ”n” (no).

```
$ rm -i pippo.c
rm: remove 'pippo.c'? y
$
```

Me opzionin ”-r” komanda **rm** fshin në mënyrë rekursive një direktori bashkë të gjitha files dhe nendirektori të përfshira në te, perveç atyre ku përdoruesi nuk ka të drejtë të beje modifikime:

```
$ rm -r src
```

Për të parë se sa hapesire ze në disk një file ose direktori, mund të përdorni komandën **du** (disk usage). Për shëmbull me komandën ”du -sk *” na shfaqet numri i KB i zene nga çdo file qe ndodhet në direktorinë aktuale. Në të kundert, për të ditur sa hapesire është akoma e lirë në çdo particion të filesystem, përdorim komandën **df** (disk free). Komanda e me poshtme shfaq hapesiren e lirë dhe të zene në çdo particion të shprehur në KB:

```
$ df -k
File system      blocchi di 1K   Usati   Dispon. Uso% Montato su
/dev/sda6          19220468  15942508    2301616  88% /
tmpfs                1033060        0    1033060    0% /lib/init/rw
```

udev	10240	248	9992	3%	/dev
tmpfs	1033060	524	1032536	1%	/dev/shm
/dev/sda3	66437944	62483816	579276	100%	/home
/dev/mmcblk0p1	3969024	313440	3655584	8%	/media/5D18-F276

Për të gjetur një file ose direktori në filesystem mund të përdorim komandën `find`, që bën të mundur gjetjen e një ose disa files duke u nisur nga emri ose nga një rregull që përcakton files që po kërkojmë. Komanda është e sofistikuar dhe bën të mundura shume veprime mbi files të gjetura. Për të parë vetëm *path* të ketyre files mund të përdorim ”`find path -name pattern -print`”. Parametrat *path* dhe *pattern* tregojnë respektivisht pozicionin nga rrenja e pemes se filesystem ku fillon kërkimi dhe rregullat që duhet të plotesojë emri i file që po kërkojmë. Për shëmbull nëse duam të gjejmë të gjitha files që përfundojnë me ”.txt” në direktorinë aktuale, përdorim komandën e me poshtme:

```
$ find . -name "*.txt" -print
./src/dati.txt
./relazione/tmp/lettera.txt
```

3.2.4 Shfaqja dhe printimi i files

Shfaqja e përbajtjes dhe printimi i file është shume i dobishëm. E njëjta gje vlen edhe për krijimin e files të reja dhe modifikimin e tyre. Kjo sidomos sepse për konfigurimin e disa karakteristikave të sistemit Linux, na duhet të modifikojmë disa files konfigurimi. Pra, mundësia e shfaqjes, printimit dhe modifikimit të files është shume e rëndësishme për përdoruesit e Linux. Për këtë qellim janë implementuar komanda të ndryshme. Ketu do paraqiten vetëm disa prej tyre, që konsiderohen të domosdoshme. Komanda e pare, dhe me elementarja, është `cat`, që shërben për të bashkuar dy ose më shume files, por mund të përdoret edhe për të shfaqur

përmbajtjen e një file teksti. Për shëmbull, komanda me poshte shfaq në terminal përmbajtjen e ”ciao.c” (një program i thjeshtë në C):

```
$ cat src/ciao.c
#include <stdlib.h>
#include <stdio.h>
int main(void) {
    printf("Ciao!\n");
    return(1);
}
$
```

Në rastet kur file është shume i gjatë dhe nuk mund të shfaqet vetëm në një ekran, rezultati do kalojë në ekran deri në rreshtin e fundit, pa na dhene mundësine ta lexojmë të plote. Për të evituar këtë problem përdoret komanda **more**, e cila ashtu si **cat** shfaq përmbajtjen e file, por e ndan në pjesë që mund të shfaqen në ekran dhe pret që përdoruesi te shtype një tast për të kaluar me radhe në pjesët e tjera të file. Komanda **less** është një version me i avancuar i **more** me disa funksione më shume. Komandat interaktive **more** dhe **less** pranojnë taste të ndryshme për të avancuar në shfaqjen e përmbajtjes se file:

Space

output avancon me një ekran;

Return

output avancon me një rresht;

b

output kthehet pas me një ekran;

q

komanda ndërpret ekzekutimin (quit);

G

shkon te fundi i file (vetëm **less**);

g

shkon te fillimi i file (vetëm less);

v

therret editorin e paracaktuar (default) të tekstit;

/

kërkon në file stringen e specifikuar menjëherë pas simbolit ”/”.

Për të shfaqur në mënyrë me komode përbajtjen e një file, mund të përdoret komanda **view** (ose ”vi -R”), me të cilën mund të spostohemi në përbajtjen e një file teksti duke përdorur tastet tipike për spostimin e kurzorit. Nëse kërkojmë të shfaqim në ekran përbajtjen e një file, që nuk është file teksti me karaktere ASCII standard (per shëmbull binary files), mund të na krijojen probleme serioze që komponentojnë sesionin e punes në terminal. Eshtë mirë që të sigurohem përllojin e file para se të kërkojmë ta shfaqim në ekran përbajtjen e tij me komandat perkatese. Për të kuptuar se me çfare lloji file kemi të bëjmë mund të përdorim komandën ”file” si me poshte:

```
$ file src/ciao.c
src/ciao.c: ASCII C program text
$ file /bin/ls
/bin/ls: ELF 32-bit MSB executable SPARC Version 1, dynamically
linked, stripped
$
```

Në rastin e parë (ciao.c) kemi të bejmë me një file teksti ASCII, përbajtja e të cilit mund të shfaqet mirë me **cat**, kurse në rastin e dyte (ls) kemi të bejmë me një file te ekzekutueshëm që nuk mund të shfaqet nepermjet komandave që pame me sipër. Komandat **head** dhe **tail** bëjnë të mundur shfaqjen respektivisht të pjesës se parë dhe të fundit të një file teksti ASCII. Zakonisht shfaqin 10 rreshtat e fillimit ose 10 rreshtat e fundit, por kjo mund të modifikohët me opsjonin ”-n”.

Komanda **wc** numeron shkronjat/simbolet, fjalët ose rreshtat në një file. Për shëmbull nëse duam të numërojmë llogarite e ndryshme të përdoruesve të sistemit përdorim:

```
$ wc -l /etc/passwd  
398 /etc/passwd
```

Perdorimi i printerit është një veprim tipik ”site dependent”, që ndryshon sipas llojit të printerit dhe filtrave të printimit të instaluar në sistem. Diferenca themelore është midis file të tekstit me karaktere ASCII, dhe file PostScript që kane një paraqitje në nivel ”tipografik” të një dokumenti ose një imazhi grafik. Nëse printeri është një printer i per gjithshëm është me e thjeshtë të printohet lloji i pare, nëse është një printer PostScript është me thjeshtë të përdoret lloji i dyte i file. Sidoqofte, si për çdo veprim tjetër në Linux, asgjë nuk është e pamundur, kështu që do jemi në gjendje të printojmë file teksti me printer PostScript dhe file PostScript me printer të per gjithshëm (generic). Komanda themelore që përdoret në këtë rast është **lpr**. Ashtu si komanda **more** edhe kjo mund të përdoret si filter për output të një programi tjetër. Kështu për shëmbull dy komandat ”**lpr pippo.c**” dhe ”**cat pippo.c — lpr**” janë ekuivalente dhe të dyja printojnë file ”**pippo.c**”. Pasi file dergohet në radhen e printimit, i shoqerohet një numer identifikativ progresiv, u quajtur *job id*. Për të parë listën e ketyre aktiviteteve që janë në pritje në radhen e printimit, duhet përdorur komanda **lpq** (line printer queue). Për të ndërprere printimin e një *job* dhe

për ta eliminuar nga lista e pritjes përdoret komanda `lprm` e shoqeruar nga *job id*. Natyrisht radha e printimit menaxhon aktivitete (jobs) nga përdorues të ndryshëm. Cdo përdorues mund të fshije nga lista vetëm aktivitetet e tij, ndërsa përdoruesi *root* mund të fshije *jobs* nga të gjithë përdoruesit e tjere.

3.2.5 Menaxhimi i proceseve

Në ambientin Linux themi ”proces” për të treguar një program në ekzekutim. Dihet që Linux është një sistem operativ multitasking, por deri tani e kemi shfrytezuar këtë karakteristike vetëm fare suportit për *multiuser*, që na krijon mundësine për ”te prekur me dore” faktin se kompjuteri, duke pasur shume përdorues të lidhur njekohësisht, po ekzekutonte në mënyrë efikase më shume se një program. Për të shfrytezuar në maksimum komoditetin e *multitasking* duhet të disponojmë një terminal grafik, që bën të mundur hapjen e disa dritareve, në të cilat mund të startojmë disa procese ne modalitet interaktiv. Kjo mund të behet edhe në një terminal alfanumerik, por edhe në këtë rast duhet një program (per shëmbull **screen**) që simulon një ambient me shume dritare. Në vijim do trajtojmë në detaje ambientin X Windows. Për momentin mjafton të themi se mund të startojmë aplikacione që avancojnë në mënyrë autonome dhe të pavarur nga njeri tjetri në një zone të ekranit të rezervuar për secilen prej tyre. Për të startuar një aplikacion, si zakonisht, duhet dhene një komande në prompt të shell. Duke vepruar në këtë mënyrë, nuk mund të startojmë njekohësisht një tjetër program në të njëjtin terminal, derisa të përfundojmë programin e pare. Ekziston një mënyrë për të shkëputur procesin ”femije” (child) nga procesi ”prind”, duke i bërë proceset të pavarur dhe autonome. Nese shtojmë simbolin ”&” (ampersand) në fund të rreshtit, komanda do të ekzekutohet në një ”sesion” të ndarë dhe mund të përdorim njekohësisht shell (ku mund të kryejmë veprime të tjera sipas nevojes) dhe programin që kemi startuar. Kjo është e mundur

me kusht që programi "femije" të mos kërkojë bashkëveprim me përdoruesin nepermjet terminalit. Shohim një shëmbull shume të thjeshte, që mund ta testojmë edhe në një terminal alfanumerik normal. Komanda "yes" shfaq një varg "y" në ekran derisa të ndërpritet nga përdoruesi. Për të ndërprere ekzekutimin e një programi duhet të shtypim Ctrl+c (break). Nëse duam ta pezullojmë perkohësisht programin duhet të shtypim Ctrl+z (stop):

```
$ yes
y
y
y
...
(1'utente batte   Ctrl-z  )
[1]+  Stopped          yes
$
```

Sistemi na informon se programi i identifikuar me job number 1, që u startua me komandën **yes**, është pezulluar perkohësisht. Komanda **jobs** shfaq listën e proceseve te startuara nga ajo shell. Për shëmbullin e meparshëm do të merrnim këtë output:

```
$ jobs
[1]+  Stopped          yes
$
```

Në këtë moment kemi dy procese aktive: *shell* dhe programin *yes*, që për momentin është i pezulluar. Në veçanti, mund të themi se procesi aktiv *shell*, zhvillohet në **foreground**, kurse procesi tjetër në **background**. Në një moment të caktuar, në seksionin aktiv të terminalit, mund të kemi vetëm një program në *foreground* dhe një ose disa programe në *background*. Për ta kaluar në *foreground* programin *yes* duhet të përdoret komanda **fg** e shoqeruar me numrin e procesit:

```
$ fg 1
```

```
y
```

```
y
```

```
y
```

```
...
```

Me sipër cituam midis njesive të një sistemi Linux, edhe device ”null” /dev/null. Tani i rikthehemë përsëri. Mund të ridrejtojmë output të një aplikacioni drejt një njesie të ndryshme nga ajo për standard output (video e terminalit) nepermjet përdorimit të simbolit ”>” (më e madhe). Provojmë të ridrejtojmë output të programit *yes* drejt device ”null”. Kjo realizohet me komandën ”yes > /dev/null”. Programi është në ekzekutim, por në ekran nuk shfaqet gje, as edhe prompt i shell për të ekzekutuar programe të tjera. Nderpresim ekzekutimin e *yes* me Ctrl+c dhe provojmë ta ristartojmë me komandën ”yes > /dev/null &”:

```
$ yes > /dev/null &
[1] 143
$ jobs
[1] 143  Running      yes >/dev/null &
$ ps
  PID TTY STAT   TIME COMMAND
    67  1  S      1:32 bash
  143  1  R      0:02 yes
  152  1  R      0:00 ps
$
```

Duke shtuar simbolin ”&” në fund të rreshtit të komandës, startojmë aplikacionin në background, duke mbajtur përdorimin e shell për të dhene komanda të tjera. Me mesazhin ”[1] 143” sistemi na komunikon se aplikacioni *yes* është procesi i parë

i startuar nga shell dhe, në tabelen e gjithë proceseve të sistemit, ka numrin 143 (pra në sistem jane 143 procese aktive). Me komandën **jobs** verifikojmë të njëjtat të dhëna dhe sistemi na komunikon se aplikacioni është në ekzekutim (running). Komanda **ps** na jep informacione mbi të gjitha proceset tona aktive, jo vetëm ato të startuara nga një shell e caktuar. Shembulli tregon se jane aktive tre procese, të gjitha në terminalin *tty1*: *bash*, aktiv prej një orë e 32 minutash dhe momentalisht i pezulluar ("S") sepse po ekzekuton komandën **ps**, që sapo u aktivua dhe është në ekzekutim ("R"), ashtu si programi *yes*. Siç përmendem me pare, çdo program aktiv në një sistem Linux ekzekutohet për emer dhe llogari të një përdoruesit të caktuar. Proseset e sistemit ekzekutohen, në pergjithësi, në emer të përdoruesit root. Edhe pse ky i fundit nuk i ka startuar nepermjet komandave, pas startimit të sistemit fillojnë ekzekutimin edhe shume procese që ngelen aktive deri në *shutdown* te sistemit. Keto procese teknikisht quhen **daemons**. Me poshte paraqitet një shëmbull i ekzekutimit të komandës **ps** në një kompjuter Sun Solaris 8:

```
$ ps -ef
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	0	0	0	Aug 16	?	0:16	sched
root	1	0	0	Aug 16	?	0:52	/etc/init -
root	2	0	0	Aug 16	?	0:00	pageout
root	3	0	0	Aug 16	?	67:57	fsflush
www	297	292	0	Aug 16	?	0:38	/usr/apache/bin/httpd
root	112	1	0	Aug 16	?	0:00	/usr/sbin/in.routed -q
root	284	1	0	Aug 16	?	0:00	/usr/lib/nfs/mountd
root	255	1	0	Aug 16	?	0:31	/usr/local/sbin/sshd
root	155	1	0	Aug 16	?	1:01	/usr/sbin/inetd -s
root	172	1	0	Aug 16	?	16:26	/usr/sbin/syslogd
www	299	292	0	Aug 16	?	0:39	/usr/apache/bin/httpd

```
nicola 10216 10214 0 21:50:29 pts/1      0:00 -tcsh
root    292      1  0  Aug 16 ?          0:04 /usr/apache/bin/httpd
...

```

Output i komandës është i organizuar në kolona dhe në rreshtin e parë tregohen etiketat që ndihmojnë për të kuptuar se çfare shpreh secila kolone. Në kolonen e parë gjejmë username (UID) te përdoruesit që po ekzekuton procesin. Në kolonen e dyte gjejmë *process id* (PID), ose numrin progresiv që identifikon në mënyrë univoke procesin. Kolona e trete përmban numrin identifikativ te procesit prind, teknikisht i quajtur *parent process id* shkurt PPID. Siç duket në shëmbullin tonë, *sched* është procesi i parë që startohet (PID=0) dhe prej tij startohen tre proceset e tjera: */etc/init* (PID=1), *pageout* (PID=2) dhe *fsflush* (PID=3). Edhe procesi *init* starton shume procese të tjera, siç duket nga shëmbulli. Eshtë interesante të veme re se prindi i procesit me ID=0 është vete procesi (PID=PPID=0). Si rregull, duhet të ketë një proces që ”lind” procesin që starton sistemin operativ dhe pastaj proceset kryesore. Nëse shohim kodet identifikuuese PID dhe PPID të procesit, duket sikur procesi ”krijon” vetveten. Ky interpretim justifikon shprehjen teknike **bootstrap**, që po ta perkthenim tekstualisht do kishte kuptimin ”te ngresh vetveten duke u terhequr nga lidheset e kepuceve”. Në kolonen TTY gjejmë terminalin nga i cili është startuar procesi. Siç tregon shëmbulli shume procese nuk kane asnje terminal, sepse jane ”daemons” të startuara gjatë fazes se nisjes se sistemit. Me komandën **kill** mund të ndërpresim me ”force” ekzekutimin e një procesi. Komanda merr si parameter edhe PID ose *job number* te procesit që duam të ndërpresim. Për shëmbull, për të ndërprire ekzekutimin e programit *yes*, dy komandat ”**kill 143**” dhe ”**kill %1**” jane ekuivalente. Komanda **kill** ka disa opsione që modifikojnë mënyrën se si ndërpritet ekzekutimi i programit. Opsiioni ”-TERM” është opsiون i paracaktuar (default) dhe e ”fton” procesin të përfundoje normalisht. Opsiioni ”-KILL” është me ”brutal” dhe ndërpitet procesin menjëherë.

3.2.6 Ndryshoret e ambientit dhe alias për komandat

Siç përmendem me pare, shell e Linux na lejon të modifikojmë ndryshoret e sesionit dhe të ambientit, në të cilat mund të ruajmë disa informacione. Zakonisht përdoren për të ruajtur disa përcakte time të bërë gjatë sesionit të punes (ndryshoret fshihen automatikisht me mbylljen e sesionit) për të bërë me komod aktivitetin tonë në Linux. Ndryshoret e sesionit vlejne vetëm në kuadrin e shell në të cilën janë përcaktuar. Nga ana tjetër, mund të përcaktojmë ndryshore ambienti, të cilat janë të vlefshme edhe për programet e startuara nga shell ku ndryshorja eshte përcaktuar. Për të përcaktuar një ndryshore sesioni duhet përdorur një sintaksë që ndryshon sipas shell që jemi duke përdorur. Në bash mjafton të shtypim ”`emer=vlerë`”, ndërsa në tesh duhet përdorur komanda `set` me sintaksën ”`set emer=vlerë`”. Edhe për përcaktimin e ndryshoreve të ambientit duhen përdorur komanda të ndryshme në varësi të shell që përdorim. Në bash komanda është `export`, ndërsa në tcsh komanda ekuivalente është `setenv`. Për shëmbull me komandën ”`export pippo=pluto`” ose ”`setenv pippo pluto`” për t'i dhene ndryshores se ambientit *pippo* vleren *pluto*. Në çdo rast, për të eliminuar një ndryshore mund të përdoret komanda `unset`, me përjashtim të ndryshoreve të ambientit në tcsh, të cilat eliminohen me komandën `unsetenv`. Për të shfaqur listën e ndryshoreve të përcaktuara të ambientit, përdoret komanda `set` pa parametra. Për të shfaqur vleren e një ndryshoreje të caktuara përdoret komanda `echo`.

```
$ pippo=pluto
$ echo $pippo
pluto
$ echo pippo
pippo
$ unset pipppo
```

\$

Një ndryshore ambienti shume e rëndësishme është ndryshorja PATH. Kur japim një komande, shell e kërkon file të ekzekutueshëm që i korrespondon komandës në direktorite e filesystem që përcaktohen në ndryshoren PATH. Kjo ndryshore përdoret për shëmbull nga komanda `which`, që gjen se në çfare direktori ndodhet fizikisht ky file i ekzekutueshëm. Ndryshorja PATH permban një varg direktorish të ndara nga simboli ”;”. Për të shtuar një direktori në këtë varg (per shëmbull `/home/nicola/bin`) përdoret në bash komanda:

```
$ export PATH=$PATH:/home/nicola/bin  
$ echo $PATH  
/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/home/nicola/bin  
$
```

Një tjetër ndryshore ambienti shume e dobishme është ajo që përcakton shenjen e prompt në rreshtin e komandave. Edhe në këtë rast sintaksa ndryshon sipas shell që jemi duke përdorur. Në bash ndryshorja quhet PS1, ndërsa në tesh quhet PROMPT. Për të përcaktuar stringen që do shfaqet su prompt mund të përdoren vargje simbolesh qe kane një kuptim specifik. Për shëmbull, shpesh përdoret një prompt që tregon username të përdoruesit, hostname të kompjuterit ku po punojmë dhe path të direktorisë aktuale (per shëmbull ”nicola@woodstock /src/perl”). Për të përcaktuar këtë prompt duhet përdorur komanda e meposhtme në bash ”

```
export PS1=\u@\h \w\$
```

”, ku ”/u” përfaqeson emrin e përdoruesit, ”/h” hostname të kompjuterit dhe ”/w” path të direktorisë aktuale. Në tcsh mund të marrim të njëjtin rezultat duke

përdorur ”setenv PROMPT ‘%n@%m %c3%#’”. Duke qene se pjesa me e madhe e komandave Linux ka shume opsione, disa nga të cilat përdoren shpesh, shell krijon mundësine për të emertuar komandat me emra alternative ose ndryshe **alias**. Komanda **alias** mund të përdoret për të zevendesuar edhe instruksione shume të gjata, si për shëmbull:

```
$ alias nproc='ps -auxw | wc -l'  
$ alias ls='ls -F'  
$ alias rm='rm -i'  
$
```

Komanda **alias** e përdorur pa parametra shfaq në ekran listën e alias të përcaktuar gjatë sesionit.

3.2.7 Të tjera komanda të dobishme

Komandat që kemi në dispozicion në një sistem Linux jane disa qindra. Perveçse është e pamundur t'i përmendim të gjitha, do ishte edhe e panevojshme, sepse shume nga komandat mund të mos i përdorni kurrë. Në këtë seksion kufizohemi duke pershkruar disa prej tyre. Nëse ka ndonje veprim që doni të kryeni, të cilen nuk e gjeni të pershkruar në këto faqe, keshilla është të shikoni manualet, sepse me siguri aty do ta gjeni një komande që ju bën pune. Shpesh kemi përdorur, në shëmbujt e meparshëm, operatoret e ridrejtimit të input/output. Me në detaje, mund të themi se simboli ”*<*” shërben për të derguar output në një file ose ne një device. Sekuenca ”*<>*” shërben për të ”ngjitur” output tek një file ekzistues, pa fshire përbajtjen e tij. Për të ridrejtuar mesazhet e gabimit gjatë një ekzekutimi, që zakonisht kalojnë në një kanal komunikimi të quajtur ”standard error”, duhet përdorur operatori ”*2<*”. Simboli ”*>*” shërben për të lexuar *standard input* nga një file ose nga një device. Në fund, simboli ”—” (pipe) shërben për të lidhur kanalin e

output të një programi me kanalin input të një programi tjetër. Në këtë mënyrë çdo gje që do të shfaqet në output gjatë ekzekutimit të programit të pare, përdoret si input për programin e dyte. Shembulli me poshtë ilustron përdorimin e operatoreve që pershkruam:

```
$ cat < lista | sort > lista.ordinata  
$
```

Programi **cat** (qe shërben për të shfaqur përbajtjen e një file) merr input nga file lista ”lista”. Output i **cat** dergohet nepermjet pipe te programi **sort** (qe rendit elementet në liste), i cili dergon output te file lista.ordinata. Në përfundim të ekzekutimit të komandës, në file lista.ordinata gjejmë elementet e ”lista”, por të renditur. Persa i perket programit **sort** duhet të shtojmë se komandat ”sort & lista & lista.ordinata” dhe ”sort lista -o lista.ordinata” do kishin të njëjtin efekt me komandën e shëmbullit të mesipërm, që kishte si qellim ilustrimin e operatoreve të ridrejtimit të I/O. Opcionet ”-r” dhe ”-n” jepin respektivisht renditjen inverse (reverse) dhe renditjen normale duke i konsideruar rreshtat si numra. Duke bashkuar në mënyrë të pershtatshme komandën **sort** me **du** dhe **head** marrim ”klasifikimin” e files që zene më shume hapesire në një direktori të caktuar, si në shëmbullin e meposhtem që shfaq tre mailbox që zene më shume hapesire në /var/mail:

```
$ du -sk /var/mail/* | sort -rn | head -3  
199904 /var/mail/root  
54712 /var/mail/andrea  
47600 /var/mail/nicola  
$
```

Ekziston një tjetër mënyrë për t’i kaluar një komande ose një programi output të një procesi tjetër: i ashtuquajturi **backtick**. Ky konsiston në thirrjen e një komande duke e mbyllur midis dy apostrofeve ‘ (backtick). Në këtë mënyrë krijohet një stringe

me output të komandës, që mund t'i kalohet si parameter një tjetër programi. Me poshte ilustrojmë përdorimin nepermjet komandave `which` dhe `file` që pershkruam me pare:

```
$ which ls
/bin/ls
$ file 'which ls'
/bin/ls: ELF 32-bit MSB executable SPARC Version 1, dynamically
linked, stripped
$
```

Një instrument shume i rëndësishëm dhe i dobishëm për të punuar me file teksti është komanda `grep`. Me këtë komande mund të kërkojmë një sekuence simbolesh në një file, pra nje stringe, të shprehur nepermjet shprehjeve të rregullta ose teknikisht ”regular expressions”. Një shprehje e rregullt është një sekuence simbolesh që përcakton një model për identifikimin e stringave. Sintaksa e përdorur është kompakte dhe në të njëjtën kohë shume e fuqishme. Do të pershkruajmë vetëm disa rregulla për formimin e shprehjeve të rregullta. Në një shprehje të rregullt çdo simbol përfaqeson vetveten, me përjashtim të disa simboleve që kane nje kuptim të veçante. Midis ketyre është pika ”.” që tregon një simbol çfaredo, ndërsa simbolet ”?”, ”+” dhe ”*” quhen kuantifikatore dhe tregojnë respektivisht se simboli ose shprehja qe kane perpara duhet të jetë në stringe zero ose një here, një ose shume here, zero ose shume here. Për të treguar një simbol me kuptim të veçante në sintaksën e shprehjeve të rregullta mjafton t'i vendosni perpara shenjen ”\” (backslash). Shprehja e rregullt me e per gjithshme është sekuanca ”.*”, e cila përfaqeson një stringe me gjatësi çfaredo (edhe bosh). Shprehja ”ab.*b.a” përfaqeson gjithë stringat që fillojnë me ”ab”, pastaj vazhdojnë me një varg (edhe bosh) simbolesh çfaredo, shkronjen ”b”, një simbol çfaredo dhe në fund shkronjen ”a”. Ketij rregulli i nenshtrohen për shëmbull stringat ”abbca”, ”abxyzwbza” dhe shume të tjera. Me `grep` mund

të shfaqim gjithë rreshtat e një file që përbajne një stringe, e cila korrespondon me modelin e shprehjes se rregullt që kemi si parameter të komandës. Opcioni ”-v” bën të mundur invertimin e perzgjedhjes: të gjithë rreshtat e file që *nuk* përbajne stringen e përfaqesuar nga shprehja e rregullt. Opcioni ”-i” e kthen kërkimin në ”case insensitive”, ose që i konsideron njesoj shkronjat e medha dhe të vogla.

```
$ grep -i 'rossi' indirizzi.txt
Mario Rossi, via marmorata, Roma, 06 9182736
$ grep -v 'Roma' indirizzi.txt
Lorenzo Bianchi, via monte bianco, Milano, 02 7349014
Giuliano Verdi, v.le beethoven, Firenze, 055 8123492
Manuela Bianchi, via monte bianco, Milano, 02 7349014
$
```

Komanda grep është shume e dobishme edhe si filter për output të programeve që shfaqin shume informacione në ekran (verbose). Për shëmbull komanda e me poshtme shfaq output të komandës ps duke përjashtuar rreshtat që përbajne stringen ”root”:

```
$ ps -ef | grep -v 'root'
UID          PID      PPID      C STIME      TTY      TIME CMD
daemon        795        1 0.0    Aug 22 ??      0:23.57 lpd Waiting
smtp       26927        1 0.0    Aug 31 ??      0:00.07 /usr/lib/sendmail
andrea     172264        1 0.0 17:19:28 ??      0:00.55 mwm -multiscreen
nicola    198161 198152 0.0 22:43:40 pts/1 0:00.08 -tcsh (tcsh)
luca      184062 184244 0.0 19:30:41 pts/2 0:00.29 pine
luca      184244 184230 0.0 19:30:38 pts/2 0:00.07 -tcsh (tcsh)
$ ps -ef | grep -v 'root' | wc -l
```

\$

Me komandën **date** sistemi shfaq datën dhe orën aktuale. Administratori i sistemit me këtë komande mund të ndryshojë datën dhe orën. Komanda **cal** pa parametra paraqet ne forme sintetike kalendarin e muajit aktual. Për të parë kalendarin e një viti të tèrë, mjafton të jepni si parameter vitin, p.sh. ”cal 2011”. Nëse përdorim dy numra si parametra, i pari interpretohet su numri i muajit dhe i dyti si viti:

```
$ cal 7 1789
July 1789
S M Tu W Th F S
      1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

\$

Shpesh na duhet të mbajmë një file në disk, edhe pse nuk e përdorim shpesh. Në këtë rast, për të kursyer vend, mund të jetë e nevojshme të **komprimojmë** këtë file. Për këtë qellim ekzistojnë disa programi komprimimi, që gjenerojnë një file në format të kompresuar, i cili ka të njëjtat informacione me originalin, por ze me pak vend ne sistem. Nëse duam të shohim informacionet, duhet të kryejmë veprimin e kundert ose **dekompresionin**. Në Linux ka shume implementime për këto funksione: *gzip*, *compress* janë me të perhapurit në këtë ambient. Me komandën ”gzip filename” komprimohet file i quajtur ”filename” në një file ”filename.gz” dhe fshihet originali. Për të kaluar përsëri te file original duhet të ekzekutojmë komandën ”gunzip filename.gz” ose ”gzip -d filename.gz”. Funksionon pak me ndryshe programi **zip**, që krijon file të komprimuar në të njëjtin format me programet PKZIP dhe WinZip.

Në një file të komprimuar mund të arkivohen shume files të tjere. Për shëmbull me komandën ”zip sorgenti.zip src/*” arkivohen në format të komprimuar, në ”sorgenti.zip”, gjithë files në direktorinë **src**. Për të marre files origjinale duhet përdorur komanda ”unzip sorgenti.zip”, ndërsa për të parë përbajtjen e ”sorgenti.zip”, pa e dekomprimuar, përdoret komanda ”unzip -v sorgenti.zip”. Komanda **tar** (tape archive) është shume i përdorur në ambientin Linux dhe zakonisht shërben për të arkivuar (pa komprimuar) shume files në një të vetëm, edhe sikur të jenë te shperndara në direktori të ndryshme. Në veçanti, është shume e përdorur për backup në shirit. Files të arkivuara me **tar** kane zgjatimin ”.tar”, ndërsa files me zgjatimin ”.tgz” janë files në format *tar* të komprimuara me *gzip*. Për të arkivuar një bashkësi files (dhe direktori) në një file në format *tar* duhet përdorur opsioni ”cvf”: c=create, f=file (pra, që vepron mbi file dhe jo në shirit), v=verbose (shfaq në ekran emrat e files që arkivohen). Për shëmbull, për të arkivuar në ”archivio.tar” gjithë permbajtjen e direktorisë aktuale përdorim komandën ”tar cvf archivio.tar .”. Për të shfaqur përbajtjen e një file të arkivuar me *tar* duhet përdorur opsioni ”tfv” (per shëmbull ”tar tfv archivio.tar”), ndërsa për të ”nxjerre” files nga arkivi duhet përdorur opsioni ”xfv” (per shëmbull ”tar xfv archivio.tar”).

```
$ ls *.txt
lettera.txt  nota_tecnica.txt      pippo.txt

$ tar cvf archivio.tar *.txt
a lettera.txt 2K
a nota_tecnica.txt 5K
a pippo.txt 1K

$ tar tfv archivio.tar
tar: blocksize = 8
-rw----- 107/60001      2159 Feb  6 16:50 2005 lettera.txt
-rw-rw-r-- 107/60001      5079 Nov 30 17:53 2000 nota_tecnica.txt
```

```
-rw-r--r-- 107/60001      762 Nov 30 17:55 2000 pippo.txt
$ tar xfv archivio.tar pippo.txt
tar: blocksize = 8
x pippo.txt, 762 bytes, 1 tape blocks
$
```

Nga rreshti i komandave në shell mund të aktivojmë një makine llogaritese të fuqishme nepermjet komandës **bc**. Behet fjalë për një ambient llogaritjesh shume me të fuqishëm se një makine llogaritese tavoline, me të cilën mund të perkufizohen edhe ndryshore dhe funksione. Për të dale nga ambienti **bc** shtypni komandën **quit** ose tastet Ctrl+d. Opsioni ”-l” specifikon përdorimin e librarise matematikore standard, prandaj rezultatet e llogaritjeve do shfaqen edhe me pjesën dhjetore.

```
$ bc -l
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type ‘warranty’.
(3+2)/2
2.500000000000000000000000
quit
$ echo "(3+2)/2" | bc -l
2.500000000000000000000000
$
```

Programi **bc** mund të përdoret edhe në modalitet jointeraktiv, duke i kaluar shprehjet që duam të llogarisim nepermjet një file që specifikohët si argument në rreshtin e komandës ose nepermjet kanalit input standard me operatorin *pipe*, si në shëmbullin e meparshëm.

3.2.8 Faqet e manualit

Burimi kryesor i informacionit mbi komandat linux dhe mbi programet e instaluara në sistem, jane faqet e manualeve ose *man pages*, që formojnë një biblioteke manualesh online. Keto faqe jane të ndara sipas argumentit në nente seksione: User command - ku gjejmë gjithë komandat e dobishme për përdoruesin; System calls - ku pershkruhen gjithë funksionet standard të gjuhes C për të thirrur funksionet e sistemit (sidomos për programatoret); Subroutines - ku pershkruhen subroutine dhe funksione të sistemit të zhvillimit (gjuha C); Devices, dove sono riportate le descrizioni dettagliate dei devices installati sul sistema; File Formats - ku gjejmë formatet e file kryesore të konfigurimit për sistemin; Games - pershkrim i lojerave të instaluara në sistem; Miscellaneous, altre descrizioni che non trovano collocazione migliore nelle altre sezioni; System administration - pershkrim i komandave për administratorin e sistemit (root).

Cdo faqe e manualit është një file dhe files që i perkasin të njëjtit seksion Jane në të njëjtën direktori.

Nëse faqja e manualit për një program të caktuar është në sistem, mund ta shohim me komandën `man` shoqeruar me emrin e programit. Shfaqja e faqe behet automatikisht me `more` dhe mund të kalojmë tekstin edhe nëse është shume i gjatë. Komanda `apropos` (ekuivalente me ”`man -k`”) na jep mundësine të kërkojmë në faqet e manualit fjalën e specifikuar si argument të komandës. Nëse do të ishim duke punuar në një X Terminal, në vend të komandës `man`, mund të përdorim komandën `xman` që na lejon të ”shfletojmë” faqet e manualit nepermjet një nderfaqesi me menu, që mund të përdoret edhe me mouse. Ad esempio con ”`man man`” si visualizzano le istruzioni për l’uso dello stesso manuale, mentre con ”`man mkdir`” si visualizza la pagina di manuale relativa al comando `mkdir`: Për shëmbull me komaden ”`man man`” shfaqen instrukzionet për përdorimin e manualit, kurse me ”`man mkdir`” shfaqet

faqja e manualit për komandën `mkdir`:

```
$ man mkdir
```

```
MKDIR(1L)
```

```
MKDIR(1L)
```

NAME

```
mkdir - make directories
```

SYNOPSIS

```
mkdir [-p] [-m mode] [--parents] [-mode=mode] [--help]
[--version] dir...
```

DESCRIPTION

This manual page documents the GNU version of `mkdir`.
`mkdir` creates a directory with each given name. By default, the mode of created directory is 0777 minus the bits set in the umask.

OPTIONS

```
-m, --mode mode
```

Set the mode of created directories to mode,

...

```
$
```

Faqet e manualit kane një format gati të ngjashëm: në fillim pershkruhet në mënyrë sintetike sintaksa e komandës (synopsis) dhe me pas listohen në mënyrë të detajuar opzionet, që mund të përdoren me komandën. Zakonisht në fund të faqes shfaqet një liste me files konfigurimi të programit (files), liste e programeve të tjera të lidhura me programin (see also) dhe probleme të njohura gjatë ekzekutimit të programit

(bugs).

3.3 Përbledhje e komandave kryesore

acroread Adobe Acrobat Reader në ambientin grafik. Shiko edhe xpdf.

alias Percakton një emer alternativ për një komande shell.

bash Bourne-Again Shell një interpretues komandash nga tastiera/file. Shiko edhe csh, ksh, sh,tcsh.

bc Makine llogaritese për terminalin alfanumerik. Shiko edhe xcalc.

bg Rifillon në sfond (background) ekzekutimin e një procesi të ndërprere me Ctrl+z.

cal Shfaq në ekran një kalendar.

calendar Shfaq gjithë takimet e planikuara dhe aktivitetet e dites dhe të dites së nesërme.

cat Bashkon files të specifikuara si argumente të komandës dhe i dergon në standard output. Shiko edhe less, more.

cd Në shell dhe në një sesion FTP shërben për të ndryshuar direktorinë aktuale.

chmod Ndryshon të drejtat për aksesin në një file.

clock Në ambientin grafik X Window shfaq një dritare me një orë. Shiko edhe datë, o'clock, xclock.

compress Komprimon dhe dekomprimon një file. Files të komprimuara me compress kane zgjatimin ”.Z”. Shiko edhe gzip, uncompress, zip.

cp Kopjon files nga një direktori në një tjetër. Shiko edhe mv, scp.

csh C Shell: është një interpretues që ekzekuton komandat nga standard input ose nga file. Shiko edhe bash, ksh, sh, tcsh.

date Shfaq/cakton orën dhe datën aktuale.

df Shfaq hapesiren e lire në filesystem. Shiko edhe du.

display Shfaq imazhet grafike në një dritare në X Window. Shiko edhe xv.

du Shfaq hapesiren e zene (ne blocks ose në KB) të direktorisë se specifikuar dhe të nenddirektive të saj. Shiko edhe df.

dvips Konverton një dokument nga formati DVI në formatin PostScript. Shiko edhe xdvi

emacs Editor për file teksti. Shiko edhe pico, vi.

exit Perfundon ekzekutimin e shell. Shiko edhe logout.

export Në Bourne Shell cakton vleren e një ndryshore ambienti. Shiko edhe setenv.

fg Sjell në foreground procesin e specifikuar me process ID ose me job number. Shiko edhe bg.

file Identifikon llojin e kodifikimit të informacioneve në një file.

find Kerkon një file në filesystem.

ftp Eshtë një klient për protokollin e transmetimit FTP (File Transfer Protocol). Shiko edhe scp.

ghostview Program për shfaqjen e një dokumenti PostScript në një dritare grafike në X Window; përdor programin GhostScript (gs).

gnuplot Program për shfaqjen e grafikeve të funksioneve me dy ndryshore; bën edhe paraqitjen grafike të të dhënave numerike nga një file teksti.

grep Kerkon një stringe në një file teksti nepermjet një shprehje të rregullt.

gs Eshtë komanda me të cilën thirret programi GhostScript, një interpretues Post-Script për shfaqjen dhe printimin e dokumenteve të atij formati.

gunzip Program për dekomprimimin e files të komprimuara me gzip.

gzip Komprimon dhe dekomprimon një file. Files të dekomprimuara me gzip kane zgjatimin ”.gz”. Shiko edhe compress, zip.

head Shfaq rreshtat e fillimit në një file teksti. Shiko edhe cat, less, more, tail.

hostname Shfaq/cakton emrin (hostname) të kompjuterit Linux.

jobs Shfaq listën e proceseve aktive të ekzekutuara në shell aktuale. Shiko edhe ps.

kill Nderpret ekzekutimin e procesit të specifikuar me Process ID ose job number.

ksh Korn Shell: është një interpretues komandash të marra nga standard input ose nga file. Shiko edhe bash, csh, sh, tcsh.

last Shfaq lidhjet e fundit të përdoruesve të sistemit.

latex Sistem për përgatitje dokumentash. Konverton një file teksti të shkruar në format LATEX në një file DVI (device independent). Shiko edhe tex, xdvi.

less Analog me more. Shiko edhe cat, head, tail, view.

logout Perfundon ekzekutimin e login shell dhe mbyll sesionin e punes të përdoruesit, duke e shkëputur nga sistemi.

lpq Shfaq radhen e printimit. Cdo element në këtë liste ka një job number dhe një përdorues perkates.

lpr Dergon një file në radhen e printimit.

lprm Heq nga radha e printimit një file.

ls Shfaq listën e files në një direktori të caktuar.

lynx Program hipertekstual për të naviguar në web (web browser). Shiko edhe mozilla, netscape, wget.

mail Program për leximin/dergimin e mesazheve të postes elektronike. Shiko edhe elm, pine.

man Shfaq faqen e manualit për komandën e caktuar. Shiko edhe xman.

mkdir Krijon direktorinë e specifikuar. Shiko edhe rmdir.

more Shfaq file të specifikuar, duke shtuar një pauze në fund të çdo ekranit. Shiko edhe cat, head, less, tail, view.

mozilla Web browser grafik për ambient X Window. Shiko edhe lynx, netscape, wget.

mv Sposton ose riemerton file të specifikuar.

netscape Web browser grafik për ambient X Window. Shiko edhe lynx, mozilla, wget.

oclock Ore për X Window. Shiko edhe clock, datë, xclock.

passwd Ndryshon password të login.

pico Editor i thjeshtë për file teksti. Shiko edhe emacs, vi.

ping Verifikon nëse një host është i arritshëm në rrjet nga kompjuteri lokal.

ps Shfaq listën e proceseve aktive. Shiko edhe jobs.

pwd Shfaq emrin e direktorisë aktuale.

rm Fshin files të specifikuara.

rmdir Fshin direktorinë e specifikuar, që duhet të jetë bosh. Shiko edhe rm.

scp Ben një kopje file në modalitet të sigurt (shifruar) midis kompjuterit lokal dhe një hosti të largët. Shiko edhe ftp, ssh.

screen Ben të mundur hapjen e shume dritareve virtuale në të njëjtin terminal alfanumerik.

set Në C Shell cakton vleren e një ndryshore. Shiko edhe setenv.

setenv Në C Shell cakton vleren e një ndryshore ambienti. Shiko edhe export, set.

sh Bourne Shell: është një interpretues që ekzekuton komandat e lexuara nga standard input ose nga file. Shiko edhe bash, csh, ksh, tcsh.

shutdown Perfundon ekzekutimin e gjithë proceseve dhe fik sistemin.

sort Rendit alfabetikisht të dhënat e një liste në një file ose në standard input.

ssh Aktivon një sesion pune në modalitet të sigurt (shifruar) drejt një serveri të largët. Shiko edhe scp, telnet.

ssh-keygen Gjeneron një kompje çelash publik dhe privat për autentikimin me çelash shifrimi asimetrik në një sesion ssh/ssh2.

tail Shfaq rreshtat e fundit të një file teksti. Shiko edhe cat, head, less, more.

talk Ben të mundur komunikimin interaktiv me një përdorues tjetër.

tar Tape Archive: Sherben për arkivimin një një file të vetëm të disa files ose direktorive.

tcs C Shell e zgjeruar: është një interpretues komandash nga standard input ose nga file. Shiko edhe bash, csh, ksh, sh.

telnet Program që shfrytezon protokollin Telnet për të përdorur një sistem të largët të lidhur në rrjet. Shiko edhe ssh.

tex Ekzekuton programin e përgatitjes tipografike të dokumentave TEX. Konverton një file teksti nga formati TEX në format DVI (device independent). Shiko edhe latex, xdv.

tin Program për leximin e lajmeve nga Usenet. Shiko edhe pine, rtin.

traceroute Shfaq itinerarin e rrjetit nga kompjuteri lokal tek një host i largët.

tty Shfaq emrin e terminalit me të cilin përdoruesi është lidhur në sistem.

uncompress Dekomprimon një file të komprimuar me compress.

unzip Dekomprimon një file të formatit .zip. Shiko edhe zip.

vi Editor për file teksti. Shiko edhe emacs, pico.

view Ekzekuton vi në modalitet leximi. Shiko edhe less, more.

w Shfaq listën e përdoruesve të lidhur në sistem. Shiko edhe finger, who.

wall Dergon një mesazh nga terminali gjithë përdoruesve të lidhur në sistem. Shiko edhe write.

wc Numeron shkronjat, fjalët ose rreshtat në një file teksti.

wget Merr një file nga një server web duke përdorur protokollin HTTP. Shiko edhe lynx, mozilla, netscape.

who Shfaq listën e përdoruesve të lidhur në sistem. Shiko edhe finger, w.

whoami Shfaq username të përdoruesit.

whois Pyet bazen e të dhënave WHOIS për të marre informacion në lidhje me subjektin që ka regjistruar një domain të caktuar në internet.

write Ben të mundur komunikimin me përdorues të tjere të lidhur në sistem, duke shfaqur në ekranin e terminalit të tyre mesazhin e shtypur. Shiko edhe wall.

xcalc Makine llogariteze në ambient X Window. Shiko edhe bc.

xclock Ora klasike për X Window. Shiko edhe clock, oclock.

xdvi Program në ambient X Window për të shfaqur dokumente në format DVI. Shiko edhe dvips, latex, tex.

xedit Editor i thjeshtë teksti në ambient X Window.

xeyes Syte që ndjekin mouse në X Window.

xhost Cakton të drejtat e aksesit në serverin grafik X Window.

xlock Screen saver me mundësine e bllokimit të terminalit në X Window. Shiko edhe xset.

xlogo Shfaq një logo të X Window në një dritare grafike.

xman Ben të mundur shfletimin e faqeve të manualit në X Window nepermjet një nderfaqesi grafik komod. Shiko edhe man.

xpdf Shfaq dhe printon dokumente në format PDF në X Window. Shiko edhe acroread.

xrdb Modifikon konfigurimin e aplikacioneve grafike në ambient X Window.

xset Vendos screen saver për sesionin e punes në ambient X Window. Shiko edhe xlock.

xsetroot Vendos disa parametra konfigurimi për ambientin X Window.

xterm Emulator terminali alfanumerik në X Window.

xv Program për shfaqjen dhe perpunimin e imazheve grafike në X Window. Shiko edhe display.

yes Shfaq pafundesisht shkronjen "y".

zip Komprimon file në formatin zip. Shiko edhe compress, gzip, unzip.

3.4 Shkarkimi

Më poshtë paraqitet një listë e plotë e elementeve që duhen shkarkuar nga rrjeti për të krijuar distribucionin, që kemi si qëllim të realizojmë. Sugjerohet që kodi të shkarkohet brenda direktorisë `/usr/src` në distribucionin pritës dhe të dekomprimohen kodet, secili në direktorinë përkatëse. Në rastin tonë është përdorur një skript në **bash** për të automatizuar të gjitha veprimet e përshkruara.

```
#!/bin/sh

for i in *.tar.gz;
{
tar xvf $i
}
for j in *.tar.bz2;
{
tar xjvf $j
}
```

Listing 3.1. `scompatta.s`

Memorizojmë këtë file me emrin **scompatta.s** dhe e bëjmë të ekzekutueshëm me komandën `chmod +x scompatta.s`. Rezultati do jetë krijimi i një direktorie për çdo file.

- Shkarko kodin e **kernel** nga faqja <http://www.kernel.org/pub/linux/kernel/>
- Shkarko kodin për **grub** nga faqja <ftp://alpha.gnu.org/gnu/grub/grub-0.97.tar.gz>
- Shkarko kodin për **bash** nga faqja <http://ftp.gnu.org/gnu/bash/>

3.5 Si?

3.6 Kernel

1. Hap direktorinë /usr/src/linux-source-{ versioni }
2. Fillo konfigurimin e kernel me një nga komandat
 - make menuconfig
 - make xconfig
 - make qconfig
3. Përfshi në kernel, jo si module, përbërësit për njojjen e pajisjeve *usb* ose *sata*
4. Përfshi në kernel përbërësit për filesystem *ext* dhe për pajisjet e input (mouse, tastiere, etj.)⁵
5. Memorizo konfigurimin e sapokrijuar
6. Ekzekuto komandën **make**
7. Ekzekuto komandën **make bzImage**
8. Ekzekuto komandën **make modules**

3.7 Hard Disk

1. Gjeni një hard disk USB të jashtëm (pendrive) me të paktën 1 GB (këshillohen 2 GB).
2. Starto Linux dhe lidh hard diskun e jashtëm (pendrive) me kompjuterin

⁵Shikoni pjesën Pse? ku ka një shembull për përzgjedhjen e moduleve

3. Futu në shell
4. Ekzekuto komandën `fdisk /dev/sda` (sdb ose sdc, etj. nëse ka pajisje të tjera USB ose SATA)
5. Krijo një particion me përmasa të përshtatshme dhe me lloj Linux (83).
6. Konfirmo ndryshimet duke i regjistruar në disk.
7. Dil nga `fdisk`
8. Ekzekuto komandën `mkfs.ext2 /dev/sda1` (mkfs.ext3) për të krijuar një filesistem në hard diskun e jashtëm (pendrive)
9. Krijo direktorine `/mnt/hd` si pikë montimi.
10. Monto HD (pendrive) të sapoformatuar me komandën `mount -t ext2 /dev/sda1 /mnt/hd`
11. Shko në direktorinë `/mnt/hd`
12. Krijo disa direktori, që do na duhen më pas, me komandën `mkdir bin boot etc dev lib sbin boot/grub lib/tls tmp var`
13. Shko në direktorinë `dev` dhe ekzekuto komandën `mknod console c 5 1`

3.8 Grub

1. Shko në direktorinë `/usr/src/grub-0.97`
2. Verifiko konfigurimin me skriptin `./configure`
3. Ekzekuto komandën `make` për komplimin e kodit
4. Verifiko nëse ekzistojnë files `stage1` dhe `stage2` në `grub`

5. Shko në direktorinë `/usr/src/grub-0.97/grub`
6. Verifiko nëse ekziston programi `grub`
7. Kopjo në direktorinë `boot/grub` të hard diskut të jashtëm (pendrive) dy files `stage1` dhe `stage2`, që ndodhen në `/usr/src/grub-0.97/{ stage1 - stage2}`
8. Kopjo files `*_stage1_5` nga `./grub-0.97/stage2` ne `/mnt/hd/boot/grub`
9. Starto `grub` me komandën `grub`, që duhet ekzekutar në `shell`
10. Ekzekuto komandën `root` (dhe me tastin TAB në tastierë shiko të gjitha mundësitet: Duhet të jene të paktën 2, një për hard diskun e jashtëm ku duam të instalojmë Linux, dhe një për hard diskun ku e kemi të instaluar Linux dhe mbi të cilin po punohet.
11. Zgjedhim hard diskun e parë, pra atë të jashtëm. Komanda do jetë diçka e ngjashme me `root (hd1,0)`
12. Ekzekuto komandën `setup (hd1)`. Vini re se nuk duhet specifikuar particioni.
13. Nëse çdo gjë është në rregull, mund të ekzekutojmë komandën `quit` për të dalë nga programi `grub`.

3.9 Bash

1. Shko në direktorinë `/usr/src/bash-3.2`
2. Ekzekuto komandën `./configure` për të verifikuar konfigurimin e sistemit
3. Ekzekuto komandën `make` për të kompluar kodin
4. Verifiko ekzistencën e programit `bash`

5. Shko në direktorinë `/usr/src/{ linux-source - versione}/arch/i386/boot`
6. Kopjo file `bzImage` që ndodhet aty në direktorinë **boot** të hard diskut të jashtëm duke i ndryshuar emrin në `vmlinuz` nëpërmjet komandës `cp bzImage /mnt/hd/boot/vmlinuz`
7. Kopjo programin **bash** në direktorinë **bin** e hard diskut të jashtëm
8. Futu në shell
9. Shko në direktorinë ku ndodhet programi bash
10. Ekzekuto komandën `ldd ./bash`
11. Do shihni në ekran një listë me libraritë e nevojshme për ekzekutimin e programit bash.
12. Kopjo të gjitha libraritë në hard diskun e jashtëm duke respektuar adresat e vendndodhjes në sistem (PATH).
13. Shko në direktorinë `/bin` të hard diskut të jashtëm
14. Krijo një link simbolik për `bash` me komandën `ln -s bash sh`

3.10 How To

1. Fik kompjuterin
2. Lidh në kompjuter hard diskun e jashtëm (pendrive)
3. Ristarto kompjuterin
4. Hyr në BIOS (zakonisht me tastet DEL ose F12 ose F2)
5. Hyr në menu Boot

6. Verifiko prezencën e hard diskut të jashtëm dhe cakto përparësinë në vargun e **boot** për të filluar nga hard disku
7. Nëse hard disku i jashtëm nuk është në listën e disqeve, vendos si pajisje për boot pajisjet e jashtme dhe pas tyre disqet.
8. Regjistro ndryshimet dhe dil nga BIOS (zakonisht me F10)
9. Prisni startimin e kompjuterit nga hard disku i jashtëm
10. Në ekran do shfaqet ambienti i komandave të **grub**
11. Ekzekuto komandën *root(hd0,0)*
12. Ekzekuto komandën **kernel** me parametrat

```
kernel /boot/vmlinuz root=/dev/sda1 rootdelay=10 init=/bin/bash
```

13. Ekzekuto komandën *boot*

3.11 Whys

3.12 Kernel

Të gjithë përbërësit e sistemit operativ Linux kanë kodin të hapur dhe mund të kon-sultohen nga kushdo që është i interesuar. Kodi i Linux kernel është i depozituar në faqen www.kernel.org. Në këtë faqe mund të merret çdo version i kernel duke filluar nga versioni 1.0 i vitit 1994 me një madhësi 1,2 MB. Për çdo version mund të gjeni kodin, rregullime (patch) të ndryshme si dhe dokumentacion sqarues. Versioni aktual është 2.6 me një madhësi prej 50MB. Administrimi i versioneve bëhet nëpërmjet shenjave të pikësimit. Rëndësia e domainit shkon nga e majta në të djathtë. Duke

filluar nga versioni aktual u vendos të shtohet një nivel duke paraqitur 4 domaine të ndryshme. Versioni i fundit do jetë 2.6.24.3. çdo gjë që ndodhet në këtë faqe e gjeni në dy variante komprimimi, **bz2** dhe **gz**. Kodi i kernel komprimohet dy herë dhe herën e parë i aplikohet një komprimim **tar**. Komprimimet janë tipike të ambientit Linux. Për një mënyrë shënimi të files është më e përshtatshme që kodi të shkarkohet dhe të dekomprimohet në direktorinë /usr/src, që zakonisht përmban kodin e shkarkuar nga përdoruesi në një filesystem Linux. Për të mësuar më shumë mbi përdorimin e komandave (si për shembull **tar**) shikoni faqen **man** të komandës.

Konfigurimi i kernel është pjesa më e rëndësishme në projektin që po paraqitet. Kernel ka disa përbërës. Disa prej tyre janë pjesë integruese dhe të domosdoshme. Disa të tjera mund të përfshihen që në fillim ose mund të shtohen në vazhdim si module gjatë kohës kur kernel është në ekzekutim. Procedura e kompilimit të kernel paraprihet nga konfigurimi i tij në një ambient çfarëdo, që shërben për këtë qëllim. Për të hapur një nga këto ambiente duhet shkuar në shell dhe nga direktoria ku kemi dekomprimuar kodin e kernel të ekzekutojmë një nga komandat e mëposhtme:

”make menuconfig”	Konfigurim me menu tekstuale.
”make xconfig”	Konfigurim i bazuar në libraritë QT.
”make gconfig”	Konfigurim i bazuar në librarite GTK+.
”make oldconfig”	Përditeson konfigurimin aktual të kernel duke përdorur file .config aktual dhe duke kërkuar konfirmim për çdo opsjon të ri që i shtohet kernel.
”make silentoldconfig”	Si oldconfig, por nuk printon asgjë në ekran përvèç rasteve kur nevojitet një përgjigje.
”make defconfig”	mënyrë konfigurimi që përdor përgjigjet në file/\$ARCH/defconfig. për gjithë opsjonet gjeneron një konfigurim të ri të kernel
”make allyesconfig”	në të cilin të gjitha opsjonet kanë përgjigjen “yes”.
”make allmodconfig”	gjeneron një konfigurim te kernel ne të cilin modulet janë të aktivizuara aty ku është e mundur.
”make allnoconfig”	gjeneron një konfigurim të ri të kernel në të cilin të gjitha opsjonet kanë përgjigjen “no”.
”make randconfig”	gjeneron një konfigurim të ri të kernel me përgjigje të rastësishme.

Në këto ambiente për çdo komponent mund të zgjidhen Y (përfshin në kernel atë komponent si pjesë integruese); M (kompilon komponentin si modul që fillimisht nuk ndodhet në kernel); N (nuk specifikon asgjë dhe nuk e kompilon). Diferenca midis ambienteve të ndryshme është vetëm në nivel grafik. Tre komandat e para të jepin një pamje në formë peme ku mund të zgjidhen opsjonet për çdo komponent. Secila nga këto ndërfaqe përdor librari të veçanta. Kjo mund të shkaktojë probleme gjatë ekzekutimit të komandave për shkak të mungesës së librarive ose të programeve të nevojshme. Në këtë rast duhet të instalohen programet e kërkuar ose mund të përdoret një komandë tjetër. Komanda e fundit jep mundësinë që të konfigurohen

komponentet me një ndërfaqe tekstuale shumë të thjeshtë dhe jo shume të modifikueshme nga përdoruesi për shkak të disa konfigurimeve të përcaktuara qysh në fillim.

Kur ekzekutohet komanda për konfigurimin, bëhen disa kontolle dhe verifikime të sistemit për të parë nëse libraritë e nevojshme për kompilimin e kernel janë në sistem. Në paragrafët e mëposhtme do paraqiten disa gabime të zakonshme për çdo lloj konfigurimi së bashku me zgjidhjet respektive.

Komanda **make** është një nga më të rëndësishmet në sistemet Linux. Në fakt kjo është komanda për kompilimin e programeve. Nëpërmjet kësaj komande mund të kompilohet një file i vetëm ose një hierarki e tërë files si në rastin e kodit te Linux. Për ekzekutimin e *make* duhen marë parasysh disa rregulla kompilimi. Këto rregulla zakonisht gjenden në *makefile* dhe në përgjithësi gjenerohen nga skripti që paraprin kompilimin dhe që emërtohet *configure*. Në kernel përveç *makefile* duhet lexuar edhe çdo gjë e shkruar në *.config*, që është një file i gjeneruar gjatë fazës së konfigurimit dhe jep informacion për çdo përbërës të kernel. Komanda *make* lexon nga *makefile* dhe nga *.config* duke kompiluar kodin C ne mënyre rekursive.

Figurat në vijim tregojnë ekzekutimin e komandave të ndryshme për konfigurimin e kernel.

Kompilimi. Komanda **make bzImage** krijon një file të ekzekutueshem te kernel që quhet **bzImage**, dhe vendoset në `/usr/src/linux-source/arch/i386/boot/` i kompresuar në formatin bz. Në kernel-at e vjetër ekzistonte vetëm formati **zImage**, por ky format kishte kapacitet të kufizuar për kernel me përmasa te mëdha, prandaj u mendua të përdorej një format më eficent. Kernel i komprimuar dekomprimohet automatikisht gjatë fazës së ekzekutimit.

Kompilimi i moduleve. Edhe kompilimi i moduleve është shumë i rëndësishëm,

3.12 – Kernel

```

File Modifica Visualizza Terminale Schede Ajuto
*
* File systems
*
Second extended fs support (EXT2_FS) [M/n/y/?] m
Ext2 extended attributes (EXT2_FS_XATTR) [y/n/?] y
Ext2 POSIX Access Control Lists (EXT2_FS_POSIX_ACL) [y/n/?] y
Ext2 Security Labels (EXT2_FS_SECURITY) [y/n/?] y
Ext2 execute_in_place support (EXT2_FS_XIP) [N/y/?] n
Ext3 extended attributes (EXT3_FS_XATTR) [M/n/y/?] m
Ext3 extended attributes (EXT3_FS_XATTR) [y/n/?] y
Ext3 POSIX Access Control Lists (EXT3_FS_POSIX_ACL) [y/n/?] y
Ext3 Security Labels (EXT3_FS_SECURITY) [y/n/?] y
JBD (ext3) debugging support (JBD_DEBUG) [N/y/?] n
ReiserFS support (REISERFS_FS) [M/n/y/?] m
Enable Reiserfs debug mode (REISERFS_FS_CHECK) [N/y/?] n
Stats in /proc/fs/reiserfs (REISERFS_PROC_INFO) [N/y/?] n
ReiserFS extended attributes (REISERFS_FS_XATTR) [y/n/?] y
ReiserFS POSIX Access Control Lists (REISERFS_FS_POSIX_ACL) [y/n/?] y
JFS file system support (JFS_FS) [M/n/y/?] m
JFS POSIX Access Control Lists (JFS_POSIX_ACL) [y/n/?] y
JFS Security Labels (JFS_SECURITY) [y/n/?] y
JFS debugging (JFS_DEBUG) [N/y/?] n
JFS statistics (JFS_STATISTICS) [N/y/?] y
XFS statistics support (XFS_STATISTICS) [N/y/?] n
XFS transaction support (XFS_TRANSACTION) [M/n/y/?] m
XFS Quota support (XFS_QUOTA) [y/n/?] y
XFS Security Label support (XFS_SECURITY) [y/n/?] y
XFS POSIX ACL support (XFS_POSIX_ACL) [y/n/?] y
XFS Realtime subvolume support (XFS_RT) [y/n/?] y
OCFS2 sysfs support (OCFS2_FS) [M/n/y/?] m
OCFS2 logging support (OCFS2_DEBUG_MASKLOG) [y/n/?] y
Minix fs support (MINIX_FS) [M/n/y/?] m
ROM file system support (ROMFS_FS) [M/n/y/?] m
Inotify file change notification support (INOTIFY) [y/n/?] y
Inotify support for userspace (INOTIFY_USER) [y/n/?] y
Quota support (QUOTA) [y/n/?] y
Old quota format support (QFMT_V1) [M/n/y/?] m
Quota format v2 support (QFMT_V2) [M/n/y/?] m
Kernel automounter support (AUTOMOUNT_FS) [M/n/y/?] m
Kernel automounter version 4 support (also supports v3) (AUTOMOUNT4_FS) [M/n/y/?] m

```

Figura 3.1. Pamja e komandës config

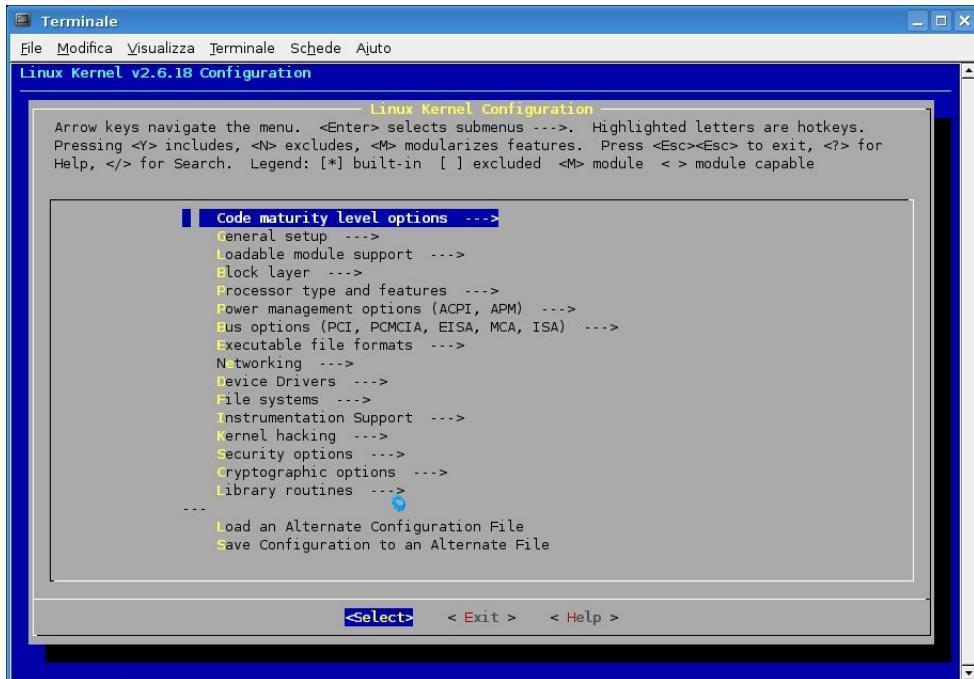


Figura 3.2. Pamja e komandës menuconfig

sepse na krijon mundësinë të shtojmë përbërësit e kernel, që kemi vendosur t'i shtojme si module gjatë ekzekutimit. Pas komplilit mund të jetë e nevojshme të

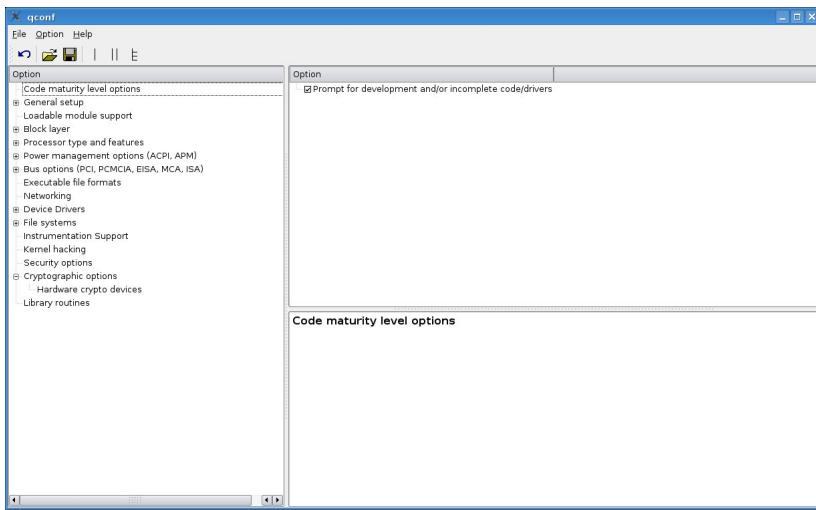


Figura 3.3. Pamja e komandës xconfig

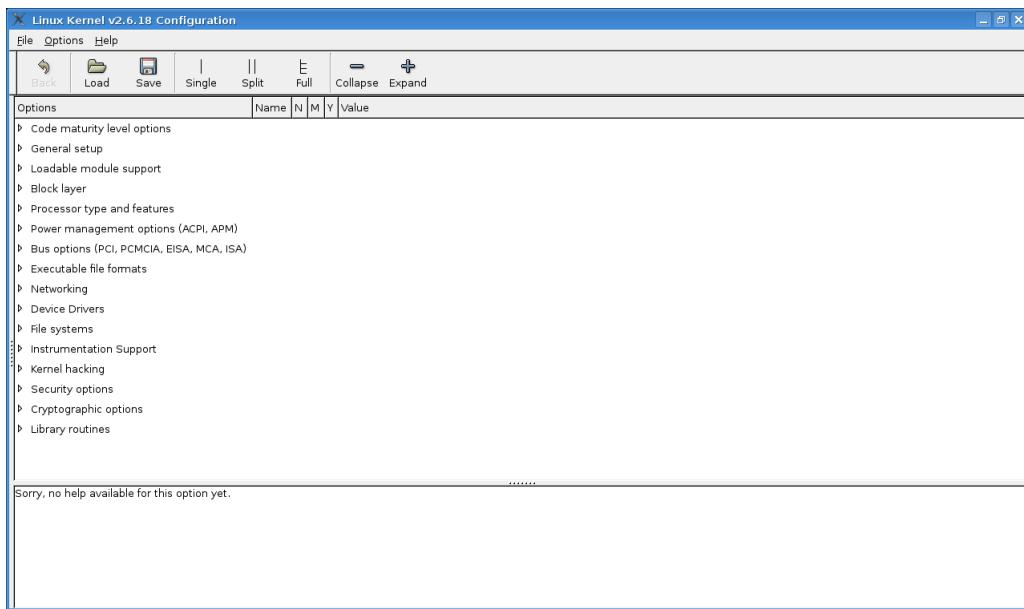


Figura 3.4. Pamja e komandës gconfig

ekzekutohet komanda *make modules install*. Rezultati i saj do jetë krijimi i direktorisë /lib/modules/{*version – i – kernel*} brenda te cilit gjëjme pemen e moduleve te kompliuara.

3.12.1 Konfigurimi

Konfigurimi i kernel është baza e gjithë punës për realizimin e distribucionit. Për këtë është e rëndësishme të vlerësohet mirë përzgjedhja e moduleve të duhura dhe të nevojshme, sipas qëllimit që kemi: përveç zgjedhjes se procesorit për optimizim të kernel, çaktivizimi i funksioneve të tepërtë ose të panevojshme, zgjedhja e pajisjeve për diskun IDE, suporti për filesystem, etj. Pra, në këtë fazë kemi disa opsione që në pamje të parë janë të fshehura, por që mund të janë shumë të rëndësishme për qëllimin përfundimtar. Nëse ndryshon kompjuteri pritës, nga portabël në fiks, mund të ndodhë që tastiera ose mouse, që më parë funksiononin, të mos funksionojnë më. Fillimisht kjo mund të duket e çuditshme, por mjafton të bëni kujdes gjatë konfigurimit të kernel. Aty ju jepet mundësia që të parashikoni edhe këto pajisje, në mënyrë që çdo gjë të funksionojë siç duhet. Me një ristartim të kernel, do të shikoni se të gjitha pajisjet do funksionojnë rregullisht. Shumë shpesh problemet si ai më sipër kanë të bëjnë me konfigurimin e kernel. Të gjitha pjesët hardware duhet të konfigurohen ashtu si duhet që të mos paraqesin probleme. Pasi arrihet një konfigurim i pranueshëm, mund të bëhet një përmirësim i kernel duke pasur parasysh këto tre pika:

- **LEHTEsim.** Kernel standard që vjen me diskun e instalimit është një model universal që përshtatet me gjithë kompjuterat dhe me të gjitha konfigurimet hardware. Prandaj ai ka një numër te madh aksesorësh që mund të janë të panevojshëm për raste specifike. Duke e rikompiluar, mund të eliminohen disa programe të panevojshme për llojin e kompjuterit që përdorni. Në këtë mënyrë hapësira e memories për kernelin do të reduktohet ndjeshëm dhe hapësira e kursyer mund të përdoret për programe të tjera.
- **OPTIMIZIM.** Kerneli mund të konfigurohet që të shfrytezojë specifikat e procesorit që përdorni, duke i bërë veprimet me te shpejta dhe me eficiente.

- PERDITESIM. Kernel i Linux përditësohet vazhdimisht sipas kërkesave për software dhe hardware. Rikompilimi shërbën për të përfshirë këto përditësimë, pa qenë nevoja të reinstalohet i gjithë sistemi.

3.12.2 Përbërësit

Zakonisht në një distro Linux përdoret një **initrd**, që i kalohet nëpërmjet Grub gjatë startimit të sistemit operativ.

Për të startuar sistemin, kernel duhet të montojë root filesystem.

Nëqoftëse përdoret një **initrd**, kernel fillimisht ngarkon këtë file dhe e monton në RAM si RAM disk.

Initrd përmban gjithë modulet e kernelit, të nevojshme për montimin e root filesystem, dhe disa skripte që ngarkojnë këto module dhe, atëherë kur është e mundur, montojnë root filesystem dhe ekzekutojnë programin **init**, që kompletion startimin e sistemit operativ.

Modulet themelore që initrd i kalon në kernel janë ata për driverat kryesore që bëjnë të mundur funksionimin e pajisjeve ku ndodhet root filesystem, për shembull pajisjet IDE, SCSI ose USB.

Me një zgjedhje projektimi, e lidhur me mundësitë e kufizuara për personalizim të initrd, është zgjedhur krijimi i një kernel me modulet e duhura në startim, pra pa përdorur një file initrd.

Ky vendim i jep një rëndësi të veçantë fazës së konfigurimit te kernel. Modulet themelore që duhen përfshirë janë ato që kanë lidhje me funksionimin e pajisjeve të memorizimit dhe modulet për filesystem.

- Seksioni Device Driver
 - SCSI device support
 - Serial Ata and Parallel Ata drivers

- Usb Support
- Seksioni Filesystems
 - Zgjidh filesystem që do përdoret (këshillohet ext3)
 - Zgjidh suportin për filesystem Windows (NTFS,VFAT)
- Seksioni Processor Type and features
 - Processor Family

Ekziston një komandë që liston gjithë pajisjet e lidhura në kompjuter nëpërmjet portave PCI. Shembulli më poshtë tregon një ekzekutim të kësaj komande dhe nxjerr në pah rëndësinë e saj.

```
debian:~# lspci
0000:00:00.0 Host bridge: Intel Corporation 915G/P/GV/GL/PL/910GL Processor to I/O Controller
(rev 0e)
0000:00:02.0 VGA compatible controller: Intel Corporation 82915G/GV/910GL Express Chipset
Family Graphics Controller (rev 0e)
0000:00:1d.0 USB Controller: Intel Corporation 82801FB/FBM/FR/fw/FRW (ICH6 Family) USB UHCI #1
(rev 04)
0000:00:1d.1 USB Controller: Intel Corporation 82801FB/FBM/FR/fw/FRW (ICH6 Family) USB UHCI #2
(rev 04)
0000:00:1d.2 USB Controller: Intel Corporation 82801FB/FBM/FR/fw/FRW (ICH6 Family) USB UHCI #3
(rev 04)
0000:00:1d.3 USB Controller: Intel Corporation 82801FB/FBM/FR/fw/FRW (ICH6 Family) USB UHCI #4
(rev 04)
0000:00:1d.7 USB Controller: Intel Corporation 82801FB/FBM/FR/fw/FRW (ICH6 Family) USB2 EHCI
Controller (rev 04)
0000:00:1e.0 PCI bridge: Intel Corporation 82801 PCI Bridge (rev d4)
0000:00:1e.2 Multimedia audio controller: Intel Corporation 82801FB/FBM/FR/fw/FRW (ICH6 Family)
AC'97 Audio Controller (rev 04)
0000:00:1f.0 ISA bridge: Intel Corporation 82801FB/FR (ICH6/ICH6R) LPC Interface Bridge (rev 04)
0000:00:1f.1 IDE interface: Intel Corporation 82801FB/FBM/FR/fw/FRW (ICH6 Family) IDE Controller
0000:00:1f.2 IDE interface: Intel Corporation 82801FB/fw (ICH6/ICH6W) SATA Controller (rev 04)
0000:00:1f.3 SMBus: Intel Corporation 82801FB/FBM/FR/fw/FRW (ICH6 Family) SMBus Controller
(rev 04)
0000:01:01.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+ (rev 10)
debian:~#
```

Rezultati i komandës jep informacion mbi zgjedhjen e moduleve që duhen përfshirë në konfigurimin e kernel.

Bus options (PCI, PCMCIA, EISA, MCA, ISA) —>

```
— PCI support
...
[*] PCI Express support

Processor type and features —>
Processor family (Pentium–Pro) —>
  ( ) 386
  ( ) 486
  ( ) 586/K5/5x86/6x86/6x86MX
  ( ) Pentium–Classic
  ( ) Pentium–MMX
  (X) Pentium–Pro

Device Drivers —>
SCSI device support —>
  <*> SCSI device support
  ...
  — SCSI support type (disk, tape, CD-ROM)
    <*> SCSI disk support
  ...
SCSI low-level drivers —>
  <*> Serial ATA (SATA) support
  ...
  <*> Intel PIIX/ICH SATA support

File systems —>
  <*> Second extended fs support
  ...
  <*> Ext3 journalling file system support
```

Lista më poshtë përmban modulet minimale që duhen përfshirë për startimin e një sistemi Linux nga disku i jashtëm USB (pen drive) me filesystem ext3

```
Processor type and features —>
Processor family (Pentium–4/Celeron (P4–based)/Pentium–4 M/older Xeon) —>

Device Drivers —>
  <*> ATA/ATAPI/MFM/RLL support —>
```

```
<*> Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support
    — Please see Documentation/ide.txt for help/info on ID
[ *] Support for SATA (deprecated; conflicts with libata)
[ *] Use old disk-only driver on primary interface
<*> Include IDE/ATA-2 DISK support
[ *] Use multi-mode by default
...
<*> SCSI emulation support
[ *] IDE ACPI support

SCSI device support —>
<*> RAID Transport Class
    — SCSI device support
<*> SCSI target support
[ *] legacy /proc/scsi/ support
    — SCSI support type (disk, tape, CD-ROM)
<*> SCSI disk support
...
<*> SCSI generic support
<*> SCSI media changer support

<*> Serial ATA (prod) and Parallel ATA (experimental) drivers —>
<*> AHCI SATA support
...
<*> Generic ATA support

[ *] USB support —>

<*> Support for Host-side USB
[ *] USB verbose debug messages
    — Miscellaneous USB options
[ *] USB device filesystem
[ *] USB device class-devices (DEPRECATED)
<*> EHCI HCD (USB 2.0) support
...
<*> USB Mass Storage support
[ *] USB Mass Storage verbose debug
[ *] Datafab Compact Flash Reader support (EXPERIMENTAL)
[ *] Freecom USB/ATAPI Bridge support
```

```

[*] ISD-200 USB/ATA Bridge support
[*] Microtech/ZiO! CompactFlash/SmartMedia support
[*] USBAT/USBAT02-based storage support (EXPERIMENTAL)
[*] SanDisk SDDR-09 (and other SmartMedia) support
[*] SanDisk SDDR-55 SmartMedia support (EXPERIMENTAL)
[*] Lexar Jumpshot Compact Flash Reader (EXPERIMENTAL)
[*] Olympus MAUSB-10/Fuji DPC-R1 support (EXPERIMENTAL) ? ?
[*] Support for Rio Karma music player
[*] The shared table of common (or usual) storage devices

```

File systems —>

...
<*> Ext3 journalling file system support

...
DOS/FAT/NT Filesystems —>

<*> VFAT (Windows-95) fs support
<*> NTFS file system support
[*] NTFS debugging support
[*] NTFS write support

3.13 Hard Disk

Për të kryer të gjitha veprimet në vijim duhet të kemi në kompjuter një version Linux çfarëdo. Duke pasur parasysh se komandat dhe programet, që u referohemi më sipër, kanë karakter të përgjithshëm, çdo version i Linux është i përshtatshëm për implementimin e projektit. Sidoqoftë, në disa raste duhen instaluar librari, programe ose paketa për të kenaqur kerkesat e software që po kompilohen.

Në sistemet Linux me **shell** zakonisht nënkuftojmë ambientin ku jepen komandat e sistemit. Gjatë viteve janë realizuar shumë lloje programesh shell. Në rastin tonë është përdorur **Bash**, që është një shell i zakonshëm ku mund të ekzekutohen komanda built-in ose komanda të jashtme.

[1 - 7]

Krijimi i partacionit. Në ambientin shell ka dy mundësi për krijimin e një particioni: njëra është tekstuale (`fdisk`), kurse tjetra grafike (`cfdisk`). Janë dy programe shumë të thjeshta me një ndihmë (*help*) relativisht të plotë. Për të kuptuar cila pajisje duhet përdorur, është e nevojshme të futet hard disku dhe të ekzekutohet komanda `mount`, që do listojë gjithë pajisjet ose burimet e montuara në filesystem. Rezultati i komandës `mount` paraqitet në formë tabelare dhe na jep mundësinë të shohim shumë informacione mbi pajisjet e montuara: pikën e montimit, llojin e pajisjes, llojin e file që i referohet pajisjes, filesystem të pajisjes dhe opsonet e montimit. Arsyja e përdorimit të komandës `mount` është verifikimi nëse ka pajisje të tjera që përdorin files per pajisjet te llojit *sd*. Nëse nuk ka, atëhere mund të ekzekutojmë komandën `cfdisk /dev/sda`. Në të kundërt, do përdorim `/dev/sdb` ose `/dev/sdc` në varësi të numrit të disqeve të të njëjtit lloj. Është e këshillueshme që particioni për sistemin kryesor të ketë përmasa të paktën 1 GB.

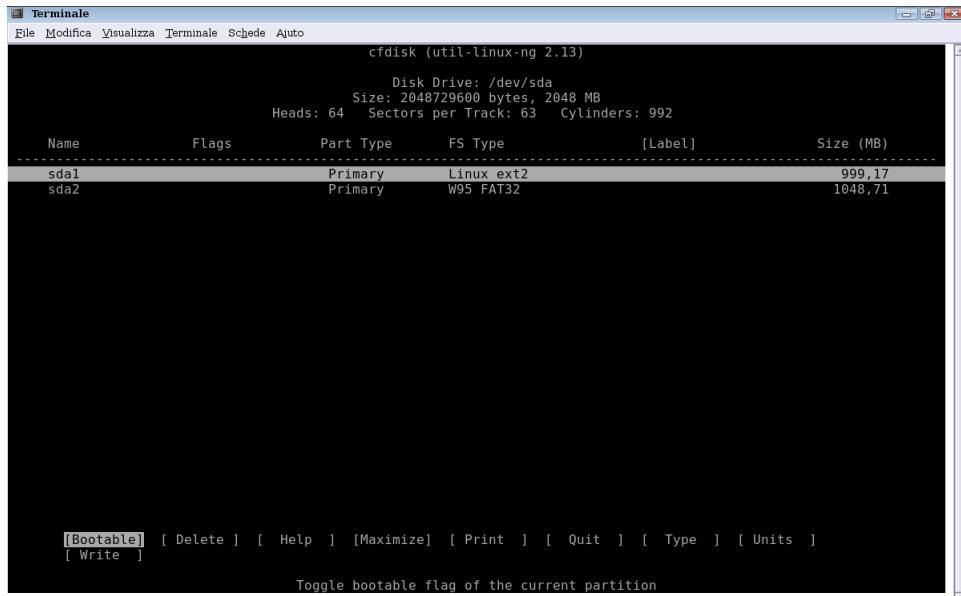
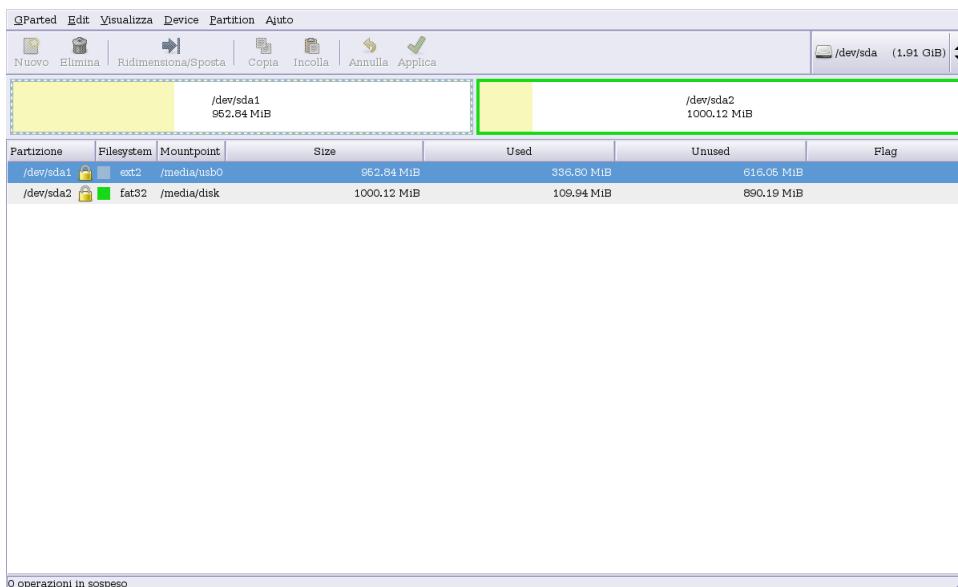


Figura 3.5. Pamje nga ekzekutimi i komandës `cfdisk`

Krijimi i filesystem. Në këtë fazë krijohet filesystem në particionin e sapopërfunduar. Zgjedhja e filesystem nuk është unique. Në fakt, për Linux ka shumë lloje filesystem të ndryshëm, secili me veçori qe i bëjnë ata shumë karakteristike. Pavarësisht nga shumëllojshmëria, këshillohet përdorimi i një filesystem i familjes **ext**, sepse ka performance më të mirë. Pra, filesystem zgjidhet midis ext2 dhe ext3. Në filesystem ext3 ofrohen disa funksione, që nuk ofrohen më parë, si për shembull journaling. Një filesystem që suporton *journaling* regjistrion ndryshimet para se t'i aplikojë në sistemin kryesor. Në këtë filesystem, mundësitet për dëmtimin e files në rast ndërprerje korrenti ose problemesh të tjera janë minimale.



Një program alternativ për sistemin e menaxhimit të particioneve është programi grafik **gparted**, një pamje të të cilit mund ta shikoni në figurën e mësipërme.

[9 - 11]

Montimi i pajisjes. Faza e montimit është shumë e rëndësishme pasi na jep mundësinë të lexojmë dhe të shkruajmë në pajisje. Kjo bëhet e mundur duke qenë se montimi parashikon lidhjen e pajisjes në pemën e filesystem të sistemit operativ. Është e përshtatshme, për shkaqe semantike, që montimi të bëhet në direktorinë /mnt. Sintaksa e komandës **mount** është e thjeshtë dhe ka një pjesë të parë ku

përcaktohet lloji i pajisjes dhe një pjesë të dytë ku përcaktohet pika e montimit në filesystem. Kjo mund të modifikohet me disa opsione që specifikojmë llojin e filesystem (-t ext2, -t vfat , -t ntfs, ...) ose të drejtat e aksesit mbi pajisjen (ro , rw).

[12]

Krijimi i pemes se filesystem. Duke përdorur komandën **mkdir** krijojen direktoritë e pemës së filesystem. Komanda ofron mundësinë e krijimit te direktive duke marë adresën absolute ose relative duke filluar nga direktoria ku ndodhemi. Në fakt, në pikën më sipër kërkohej ndryshimi i direktorisë për të mos rënduar sintaksën e komandës. Disa direktori për momentin janë bosh, të tjera do popullohen sipas nevojave.

[13]

Krijimi i console. Komanda **mknod** përdoret për të krijuar disa file speciale si file me karaktere ose me bloqe. Në rastin tonë po krijojmë një file special me karaktere te llojit console. Opzioni *c* i referohet një file me karaktere dhe dy numrat më pas janë numri primar dhe sekondar i pajisjes. Mund të deklarohen deri në 63 console duke ndryshuar numrin sekondar në vlerat nga 1 deri në 63. Për momentin console është e vetmja pajisje që na duhet për të shfaqur mesazhet ne video.

3.13.1 Probleme

Arkivat për pajisjet (device files) janë file të vecanta që sistemet Unix-like i asociojnë me pajisjet fizike. Pjesa më e madhe e tyre krijojen gjatë instalimit te distribucionit. Lista e këtyre files për pajisjet që suportohen nga kerneli Linux mund të shikohet ne /usr/src/linux/Documentation/devices.txt. Një file pajisje karakterizohet nga:

- - emri - është emri i file që identifikon pajisjen dhe nuk ka asnë ndikim mbi funksionimin e pajisjes;
- - lloji - është lloji i pajisjes. Mund te jete me bloqe (hard disk, CD-ROM, ...)

ose me karaktere (terminal, njësi me shirita, ...);

- - numri major - tregon driver për funksionimin e pajisjes;
- - numri minor - identifikon pajisjen e driver;

Në rastet kur është e nevojshme të krijohet një file i ri për pajisje, mund të përdoret skripti MAKEDEV që gjendet në /dev/MAKEDEV ose brenda direktorisë /sbin. Këto file për pajisjet shërbejnë si driver për sistemin dhe bëjnë të mundur menaxhimin e mekanizmave për komunikim me një pajisje të veçantë. Në këtë kontekst, nëse provohet startimi i distribucionit brenda një particioni të kompjuterit, pa përdorur një hard disk të jashtëm, vihet re mungesa e files për pajisjet, edhe nëse me skriptin MAKEDEV u krijuan ne /dev disa files sd*, tashmë duhen file të tjerë me parashtesën hd*. Lista e partacioneve të njoitura nga sistemi gjendet në /proc/partitions.

3.14 BootLoader - Grub

BootLoader është një program i domosdoshëm për startimin e sistemit operativ Linux. Rëndësia e tij qëndron në faktin se ai jep mundësinë për të percaktuar files që ekzekutohen dhe për përcaktimin e opsiioneve për ekzekutimin. Përveç kesaj është e mundur të përcaktohet edhe pajisja (hard disku) ku janë të vendosur këta files. Në botën Linux ka disa programe bootloader të ndryshëm nga njëri tjetri. Në këtë projekt është zgjedhur grub, sepse është më i përdoruri dhe thjeshton punën për realizimin e projektit.

[1 - 3]

Kompilimi i programeve në Linux zakonisht paraprihet nga konfigurimi, që bëhet nëpërmjet ekzekutimit të një skripti. Objektivi kryesor i këtij skripti është të verifikojë nëse sistemi mund ta ekzekutojë me sukses kompilimin. Në këtë fazë, ne fakt,

verifikohet prezenca në sistem e programeve dhe librarive të nevojshme për kompilimin. Ekzekutimi i skriptit para kompilimit shërben edhe për të gjetur problemet eventuale në sistem, sepse skripti është projektuar për një bllokim të menjëherëshëm në rast problemesh. Ky bllokim i jep mundësinë përdoruesit të kuqtojë se ku qendron problemi dhe ta rregullojë sipas rastit. Rezultati i ekzekutimit te skriptit është krijimi i **makefile**, mungesa e të cilit e bën të pamundur ekzekutimin e kompilimit me komandën **make**. Siç u përmend më parë, komanda make kompilon në mënyrë rekursive të gjitha files që gjen. Për këtë arsyе duhet të vendosemi në nëndirektorinë /grub për të gjetur programin grub të sapokompiluar.

[4 - 8]

Faza e instalimit është përgjithësisht atipike. Kjo parregullsi vjen nga fakti që instalimi bëhet në një kompjuter të ndryshëm nga ai ku është krijuar distribucioni. Faza e parë parashikon kopjin e disa files të ekzekutueshëm nga direktoria origjinale në direktorine /boot/grub te hard diskut destinacion. Këto emra të files kanë një kuptim të qartë dhe u referohet stadeve që përbëjnë fazën e startimit të sistemit. Po të shikoni me kujdes mund të dalloni se file *stage1* ka madhësinë 512 byte, pra një sektor: kjo është pjesa e parë e kodit që përdoret gjatë startimit. File *stage2* përfaqëson kodin që nevojitet për kompletimin e startimit, kurse files të serië *-*stage1_5* përfaqesojnë një faze të ndërmjetme që mund të futet, nëse është e nevojshme, menjëherë para *stage2*. Mund të dallohet sesi files te ekzekutueshme, me prapashtesen *_stage1_5*, kanë si parashtesë emrin e filesystem. Pra, në bazë të filesystem që kemi në hard diskun destinacion duhet te zgjidhen edhe keto files. Këshillohet të kopjohen ato për windows filesystems (vfat dhe ntfs) si dhe ato Linux.

[9 - 14]

Në ambientin grub mund të ekzekutohen disa komanda tipike. Për të gjetur listen e ketyre komandave mund të përdorni tastin TAB. Një funksion interesant i ambientit grub është kompletimi automatik i komandave ose i parametrave të komandave.

Nëpërmjet tastit TAB ofrohet mundësia për kompletimin e komandave dhe e vlerave në input. Për shembull, komanda *root* merr si input diskun dhe particionin që do bëhet root. Pasi shkruhet komanda dhe hapet kllapa, duke shtypur tastin TAB shfaqet një listë e plotë e hard disks që ndodhen në kompjuter, të particioneve në secilin hard disk dhe të filesystem të çdo particioni. Në këtë menyre bëhet më i lehtë dhe më i kuq tueshëm administrimi i tyre. Në rast ambiguiteti mund të përdoret komanda `find /boot/grub/stage1` për të shfaqur listën e particioneve të mundshme.

```

File Modifica Visualizza Terminale Schede Ajuto

GNU GRUB version 0.97 (640K lower / 3072K upper memory)

[ Minimal BASH-like line editing is supported. For
the first word, TAB lists possible command
completions. Anywhere else TAB lists the possible
completions of a device/filename. ]

grub> root (hd
      Possible disks are: hd0 hd1

grub> root (hd1,
      Possible partitions are:
      Partition num: 0,   Filesystem type is ext2fs, partition type 0x83
      Partition num: 1,   Filesystem type is fat, partition type 0xb

grub> root (hd1,0)
      Filesystem type is ext2fs, partition type 0x83

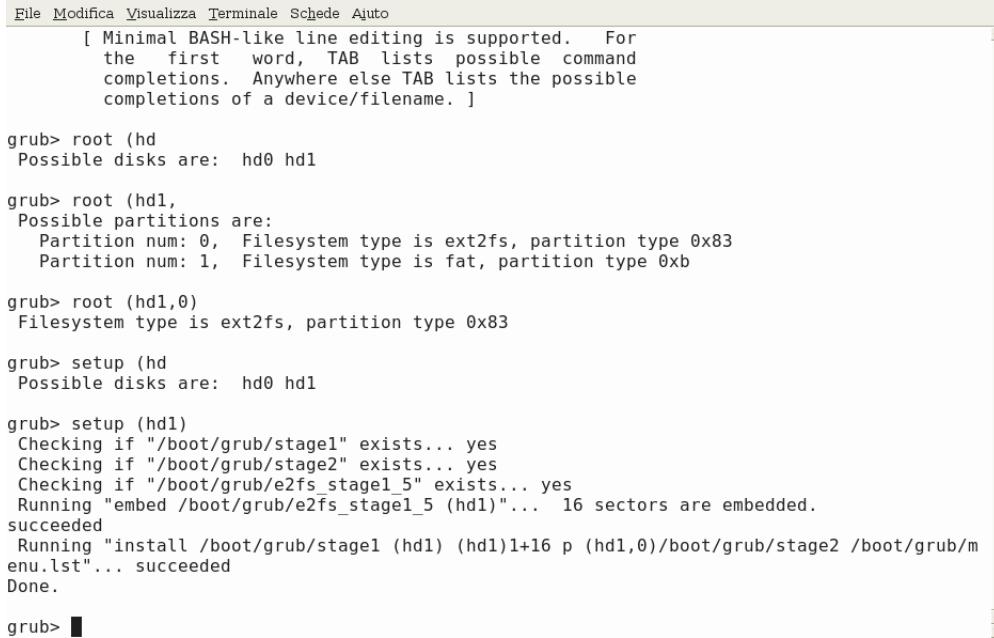
grub> ■

```

Figura 3.6. Pamje nga ekzekutimi i komandës `root` në ambientin `grub`

Ashtu si thamë më sipër, komanda *root* shërben për të përcaktuar cili particion permban files per startimin, kurse komanda *setup* merr identifikativin e diskut ku Master Boot Record (MBR) do të instalojë bootloader. Gjatë ekzekutimit të komandës *setup* do verifikohet prezenca e files që janë të domosdoshme për instalimin korrekt të grub. Eventualisht mund të zgjerohet grub duke perdorur disa file konfigurimi që vendosen në direktorinë `/boot/grub`. Një nga këto file është edhe `menu.lst`.

Në këtë file mund të krijohet dhe të personalizohet një menu tekstuale që thjeshton komandat duke dhënë mundësinë e vendosjes së grupeve të komandave me opsiione (root - kernel - boot).



```

File Modifica Visualizza Terminale Schede Ajuto
[ Minimal BASH-like line editing is supported. For
the first word, TAB lists possible command
completions. Anywhere else TAB lists the possible
completions of a device/filename. ]

grub> root (hd
      Possible disks are: hd0 hd1

grub> root (hd1,
      Possible partitions are:
      Partition num: 0, Filesystem type is ext2fs, partition type 0x83
      Partition num: 1, Filesystem type is fat, partition type 0xb

grub> root (hd1,0)
      Filesystem type is ext2fs, partition type 0x83

grub> setup (hd
      Possible disks are: hd0 hd1

grub> setup (hd1)
      Checking if "/boot/grub/stage1" exists... yes
      Checking if "/boot/grub/stage2" exists... yes
      Checking if "/boot/grub/e2fs_stage1_5" exists... yes
      Running "embed /boot/grub/e2fs_stage1_5 (hd1)"... 16 sectors are embedded.
      succeeded
      Running "install /boot/grub/stage1 (hd1) (hd1)1+16 p (hd1,0)/boot/grub/stage2 /boot/grub/m
enu.lst"... succeeded
      Done.

grub> ■

```

Figura 3.7. Pamje nga ekzekutimi i komandës `setup` në ambientin `grub`

3.15 Bash

Instalimi i kernel. Në këtë faze, kerneli i kompiluar do kopjohet në hard disk. Për motive historike dhe semantike është e përshtatshme që emërtimi `bzImage` të ndryshohet në `vmlinuz`. File `bzImage` ndodhet në direktorinë `/usr/src/{ linux-source-version }/arch/i386/boot`, kurse kopja duhet vendosur në `/mnt/hd/boot/`.

Instalimi i `bash` Çdo file i ekzekutueshëm në ambientin Linux ka nevoje për disa librari që të ekzekutohet normalisht. Me komandën `ldd` marrim në output një

listë me të gjitha libraritë e nevojshme për ekzekutimin e një file. Lista përfshin edhe vendndodhjen fizike të librarive në filesystem. Pra, për të instaluar *bash* ose programe të tjera të ekzekutueshme nga disku origjinë në atë destinacion, duhen kopjuar edhe libraritë e nevojshme duke respektuar vendndodhjet në sistem, sipas të dhënave që na jep komanda *ldd*. Bie në sy se shumë librari janë të përsëritura, meqë një pjesë e tyre përdoren nga disa programe. Përveç kësaj, bie në sy se libraritë vendoset në */lib* dhe */usr/lib*.

Në rastin tonë, duke ekzekutuar komandën **ldd bash** në shell të distribucionit origjinë do marrim si rezultat listën e librarive dhe vendndodhjen e tyre në filesystem.

```
/usr/src/bash-3.2... ldd ./bash
    linux-gate.so.1 => (0xfffffe000 )
    libncurses.so.5 => /lib/libncurses.so.5 (0xb7f68000)
    libdl.so.2 => /lib/i686/cmov/libdl.so.2 (0xb7f64000)
    libc.so.6 => /lib/i686/cmov/libc.so.6 (0xb7e1b000)
    /lib/ld-linux.so.2 (0xb7fad000)
/usr/src/bash-3.2...
```

Rezultati i komandës shfaqet në dy kolona, që tregojnë respektivisht emrin e librarisë dhe vendndodhjen fizike të saj në filesystem. Adresa fizike shenohet me =>. Gjatë kopjimit të librarive duhet respektuar adresa e tyre në filesystem. I vetmi përjashtim është me librarinë *linux-gate.so.1*, që nuk ka si referim asnjë file fizik, por një pozicion në memorie. Kjo librari në fakt krijohet në mënyrë dinamike (duke filluar nga versioni 2.6) dhe përdoret për të menaxhuar interrupts. Gjatë startimit kernel njeh procesorin dhe krijon librarinë për të optimizuar menaxhimin e *system calls*.

3.16 Test

Bios (Basic Input Output System) është programi i parë që ekzekutohet nga një kompjuter personal. Funksionet e tij janë të lidhura me hardware. Në menunë e Bios është e mundur të ndryshohet ora dhe data e sistemit, të verifikohet prezenca dhe funksionimi korrekt i pajisjeve hardware të instaluar, të vendoset se në cilën pajisje ndodhet kodi i sistemit operativ. Në rastin tonë, nëpërmjet Bios modifikojmë pajisjen ku ndodhet sistemi operativ që do startohet. Hard disqe të jashtëm njihen nga Bios ndërmjet pajisjeve të jashtme, por duke qenë se ka shumë versione Bios, ndodh që hard diskun ta gjeni në listën e pajisjeve fikse. Duke qenë se shpesh në kompjuter mund të ndodhen shumë disqe, Bios krijon mundësinë e zgjedhjes duke përcaktuar një prioritet përfazën e startimit. Në rastin tonë hard diskut ku ndodhet distribucioni i caktohet prioriteti maksimal. Në Bios ekziston mundësia që të aplikohen ndryshimet e bëra ose të kalohet në gjendjen fillestare duke evituar probleme të mundshme.

Pasi startohet kompjuteri nga hard disku i jashtëm na shfaqet ambienti grub. Kjo ndodh sepse akoma nuk kemi përcaktuar se si duhet të veprohet, pra nuk kemi ndryshuar file *menu.lst*. Ekzekutojmë komandën root (hd0,0). Si ndihmë mund të përdorni tastin Tab për te parë listën e disqeve që njihen nga grub. Në rastin tonë nuk ka dyshime sepse *hd0* është gjithmonë hard disku i startimit, pra ai i jashtëm. Ekzekutojmë komandën *kernel*. Nëpërmjet kësaj komande përcaktohet me saktësi cili do jetë file që ekzekutohet gjatë startimit dhe caktohen disa parametra themelore për startimin korrekt të sistemit operativ. Për këtë duhet të japim adresën (path) e plotë ë vendndodhjes përfiile /boot/vmlinuz. Opcionet e nevojshme gjatë kësaj faze janë root, rootdelay dhe init.

Opsioni **root**. Pas ekzekutimit të kernel dhe njohjes nga ana e tij të të gjithë pajisjeve të instaluar në kompjuter, kernel kalon ekzekutimin në një particion të

hard diskut. Për të specifikuar se cili është particioni dhe cili është hard disku i referimit duhen kaluar parametrat te opzioni *root*=, duke specifikuar pajisjen dhe particionin. Vini re se hard disku dhe particioni identifikohen sikur ambienti Linux të ishte në punë. Në veçanti, duhet specifikuar direktoria ku ndodhet file i pajisjes dhe emri i file. Sintaksa është e thjeshtë: specifikohet prefiksi që tregon tipologjinë e hard diskut, vazhdon me një shkronjë që tregon hierarkine midis pajisjeve të të njëjtit lloj. Në fund vendoset particioni nëpërmjet një numri. Për të kuptuar më mirë japim disa shembuj:

- /dev/hda1

Particion në hard diskun e parë të llojit eide/scsi.

- /dev/hda4

Particioni i katërt në hard diskun e pare të llojit eide/scsi.

- /dev/hdb1

Particioni i parë në hard diskun e dytë të llojit eide/scsi.

- /dev/sda1

Particioni i parë në hard diskun e parë të llojit sata/usb.

- /dev/sdc6

Particioni i gjashtë në hard diskun e tretë të llojit sata/usb.

Duke filluar nga versioni 2.6.22 u implementuan disa funksione të tjera. Për menaxhimin e pajisjeve periferike usb janë shtuar device uba. Ky funksion krijon mundësinë për të dalluar hard diskun sata nga pajisjet periferike usb. Kjo sjell avantazhe, për shembull në ambientin grub sepse komanda kernel bëhet unike për të gjitha makinanat.

```
kernel /boot/vmlinuz root=/dev/uba1 rootdelay=10 init=/bin/bash
```

gjë që nuk ishte e mundur më parë sepse duhej personalizuar në bazë të numrit të disqeve.

3.16.1 Konfigurimi

Opsioni **rootdelay** shërben për të përcaktuar numrin e sekondave të pritjes para se të kërkohet root nga ana e kernel. Ky opzioni është i rëndësishëm për kernel të bazuar në një hard disk usb, sepse nevojiten disa sekonda që pajisja të njihet. Një kohë e arsyeshme për këtë vonesë është të paktën 10 sekonda.

Opsioni **init**. Pasi ngarkohet kernel në memorie dhe pasi njihet filesystem i partitionit root, është e nevojshme të specifikohet se cili do jetë procesi fillestar. Kjo bëhet nëpërmjet opzionit *init*. Në distribucionet Linux klasike ky është një proces themelor sepse është përgjegjës për startimin e proceseve daemon, për ngarkimin në memorie të moduleve të kernel, për startimin e skripteve të konfigurimit, për startimin e serverit grafik ose te serverave të tjera të nevojshëm për mirëfunksionimin e sistemit operativ. Në rastin në fjalë, duke qenë se nuk është instaluar asnjë program, është e mjaftueshme të specifikohet si proces fillestar programi bash, që ndodhet në direktorinë /bin për të startuar sistemin me ambientin shell të bash.

Pasi ekzekutohet kjo komandë, grub merr kernel nga filesystem dhe e ngarkon në memorie, duke verifikuar nëse ka probleme për mungesë të files që nevojiten. Me komandën boot ekzekutohet ky file dhe starton efektivisht sistemi operativ. Në fund të këtij procesi duhet të shfaqet ambienti bash. Mund të verifikohet funksionimi i ketij ambienti me ekzekutimin e komandave minimale (echo, eval, etj.). Mund të verifikojme gjithashtu se nuk janë akoma të disponueshme komandat ls, cd, etj. Procedura për instalimin e bash do të përsëritet në mënyrë të ngjashme edhe për komandat e tjera te shell.

3.17 FAQ

3.18 Kernel

Sinjalizoni problemet dhe pyetjet e mundshme

- Çfare është ”Kernel Panic”?

Një situatë ”kernel panic” paraqitet në rast se sistemi operativ gjen një gabim të ”pandreqshëm”, që con në mënyrë të pakthyeshme në ristartimin/bllokimin e sistemit. Procedura e kernel që menaxhon një situatë të tillë ”paniku” njihet me emrin `panic()` dhe është projektuar për të shfaqur një mesazh në console, për të kopjuar në disk një imazh të memories së kernel dhe për të lejuar në vazhdim një *debugging*. Në vijim procedura pret një ristartim manual të sistemit. Shumë shpesh situatat ”kernel panic” krijohen nga tentativat e sistemit operativ për të lexuar një adrese memorie jo të vlefshme ose të paautorizuar. Në raste të tjera, situata krijohet nga një gabim në nivel hardware ose nga ndonjë ”bug” në sistemin operativ. Kur Linux nuk di si të veproje, si në këtë rast, ai shfaq një mesazh ”kernel panic” dhe ndalon. Sidoqoftë, edhe në rast ”paniku”, sistemi ndalon në mënyrë ”elegante”. Kërkon të shkruajë në disk të gjitha ato që duhen regjistruar (veprim i quajtur ”syncing”), dhe nëse veprimi përfundon me sukses shfaqet mesazhi ”not syncing”. Duket qartë se mesazhi është çorientues, sepse në fakt nuk është ky motivi i ”panikut”. Shumë shpesh në rastet e ”panikut” shfaqet mesazhi ”tried to kill init”. Në të vërtetë, mesazhi na tregon se procesi `init` po ”vdes”, gjë që nuk është e pranueshme sepse `init` është një program i veçantë në Linux, që hyn në punë kur sistemi startohet.

- `make menuconfig`

pasi ekzekutohet komanda, rezultati fillon me

```
...
HOSTCC scripts/kconfig/lxdialog/checklist.o
In file included from scripts/kconfig/lxdialog/checklist.c:24:
scripts/kconfig/lxdialog/dialog.h:32:20:
      error: curses.h: No file or directory
...
```

Zgjidhje e mundshme

instalohet libraria *curses* duke përdorur menaxherin e paketave (libraria aktuale është *libncurses5-dev*)

- **make**

në fazën e kompilimit procedura ndërpritet me

```
...
kernel/built-in.o: In function `timespec_add_ns':
/usr/src/linux-2.6.23/include/linux/time.h:177: undefined reference
      to `__umoddi3'
/usr/src/linux-2.6.23/include/linux/time.h:177: undefined reference
      to `__udivdi3'
kernel/built-in.o: In function `timespec_add_ns':
/usr/src/linux-2.6.23/kernel/time/timekeeping.c:127: undefined reference
      to `__udivdi3'
/usr/src/linux-2.6.23/kernel/time/timekeeping.c:127: undefined reference
      to `__umoddi3'
make: *** [.tmp_vmlinux1] Error 1
```

Zgjidhje e mundshme

Verifikoni versionin e kompilatorit *gcc*. Ndonjëherë është e nevojshme të përdoret një version i përshtatur me versionin e kernel. Në rastin tonë, meqë kernel është version 2.6.23 nuk duhet përdorur versionin *gcc-4.3*, por një version më i vjetër.

Pas instalimit të *gcc* duhen këto komanda:

```
cd /usr/bin  
rm gcc (eliminon linkun simbolik për versionin që do përdoret)  
ln -s gcc-3.4 gcc (ose për versionin që keni instaluar) (rikrijon linkun simbolik)
```

3.19 Hard disk

- Të gjitha veprimet e modifikimit të tabelës së particioneve dhe të krijimit të filesystem të ri behen duke pasur të drejta administratori. Për këtë është e nevojshme

```
sudo sh  
cfdisk ...
```

për të krijuar një shell administratori ku të kryhen të gjitha veprimet me të drejtat e root

ose

```
sudo cfdisk ...  
për të ekzekutuar komandën cfdisk si administrator.
```

3.20 Grub

Pasi starton programi grub dhe ekzekutohet komanda `root (hd1,0)`, nuk është e mundur të instalohet boot loader në hard disk (kemi vetëm një hard disk hd0 ose vetëm fd0)

Zgjidhje e mundshme

Gjatë fazës së instalimit, grub modifikon MBR të hard diskut. Ky veprim u lejohet vetëm përdoruesve të privilegjuar. Ekzekutoni komandën me të drejta super user

duke përdorur `sudo grub`

3.21 Bash

Gjatë startimit sistemi kalon në *Kernel Panic* sepse nuk arrin të ekzekutojë `/bin/bash`

Zgjidhje e mundshme

Problemi zakonisht shkaktohet nga mungesa e ndonjë librarie që nevojitet për ekzekutimin korrekt. Ekzekutoni komandën `ldd /bin/bash` dhe verifikoni të gjitha libraritë e nevojshme sipas listës.

3.22 Coreutils

3.22.1 Shkarkimi

- Shkarkoni versionin e fundit nga site <ftp://ftp.gnu.org/gnu/coreutils/> aktualisht 6.9
- Shkarkoni nga <http://sourceforge.net/projects/e2fsprogs/> paketën që përmban utilite të mbi filesystem.
- Shkarkoni nga site <ftp://ftp.win.tue.nl/pub/linux-local/utils/util-linux/> versionin e fundit të paketës util-Linux

3.22.2 Si?

1. Hyni në direktorinë /usr/src/coreutils-6.9
2. Ekzekutoni komandën nga shell./configure
3. Ekzekutoni komandën *make*
4. Hyni në nëndirektorinë *src*
5. Në këtë kartelë është e mundur të gjeni kodin e komandave më të përdorura në shell, kodin objekt të kompilimit dhe të ekzekutueshmit të sapokrijuar.
6. Kopjoni më të rëndësishmet dhe më të zakonshmet nga kjo kartelë tek direktoria bin e hard diskut. Kjo mund të bëhet me komandën:
*cp cat chgrp chmod chown cp date dd df /mnt/hd/bin
cp hostname ln ls mkdir mkfifo mknod /mnt/hd/bin
cp mv rm rmdir stty su sync uname /mnt/hd/bin*

7. Verifikoni nëse nevojiten librari të tjera nëpërmjet komandës *ldd* siç është bërë për *bash*
8. Kopjoni libraritë e nevojshme duke respektuar të njëjtin PATH.

3.23 e2fsprogs

1. Hyni në direktorinë e2fsprogs që përmban kodet.
2. Konfiguroni ndryshoren e ambientit me *export CC=“gcc -mcpu=i386”*
3. Ekzekutoni komandën *./configure --host=i386-pc-linux-gnu*
4. Nisni kompilacionin nëpërmjet komandës *make*
5. Hyni në direktorinë e2fsck
6. Ekzekutoni komandën *cp e2fsck.shared /mnt/hd/sbin/e2fsck*
7. Hyni në direktorinë/misc
8. Ekzekutoni komandën *cp fsck mke2fs /mnt/hd/sbin*
9. Hyni në kartelë /mnt/hd/sbin
10. Krijoni link simbolik nëpërmjet komandave
 - *ln -s e2fsck fsck.ext2*
 - *ln -s mke2fs mkfs.ext2*

3.24 Util-Linux

1. Hyni në direktorinë /usr/src/{util-Linux-versione}

2. Ekzekutoni komandën `./configure`
3. Ekzekutoni komandën `make`
4. Kopjoni të ekzekutueshmit e sapokrijuar tek hard disk i jashtëm nëpërmjet komandave
 - `cp disk-utils/mkfs /mnt/hd/sbin`
 - `cp fdisk/fdisk /mnt/hd/sbin`
 - `cp login-utils/agetty /mnt/hd/sbin`
 - `ln -s agetty /mnt/hd/sbin/getty`
 - `cp login-utils/login /mnt/hd/bin`
 - `cp misc-utils/kill /mnt/hd/bin`
 - `cp mount/mount /mnt/hd/bin`
 - `cp mount/umount /mnt/hd/bin`
 - `cp mount/swapon /mnt/hd/sbin`
 - `cp sys-utils/dmesg /mnt/hd/bin`
5. Verifikoni nëse nevojitet të kopjoni librari nëpërmjet komandës `ldd`
6. Hyni në direktorinë `/mnt/hd/dev`
7. Krijoni file dispozitiv
 - `mknod null c 1 3`
 - `mknod fd0 b 2 0`
 - `mknod ram0 b 1 0`
8. Hyni në direktorinë `/mnt/hd/etc`

9. Duke përdorur një editor teksti si *vi* ose *emacs* krijoni file e mëposhtëm *fstab*

proc	\proc	proc	defaults	0	0
\dev\sda1	\	ext2	defaults	1	1

Listing 3.2. /etc/fstab

10. Krijoni një file bosh gjithmonë në direktorinë */mnt/hd/etc* me komandën

echo -n > mtab

11. Ekzekutoni komandën *mkdir /mnt/hd/etc/init.d*

12. Krijoni file si më poshtë në këtë kartelë dhe riemërtojeni *local_fs*

```
#!/bin/sh
# local-fs - check and mount local filesystems
#
PATH=/sbin:/bin ; export PATH

fsck -ATCp
if [ $? -gt 1 ]; then
    echo "Errori presenti nel filesystem! Si richiede intervento manuale."
    /bin/sh
else
    echo "Rimontaggio di / in modalit\`a lettura-scrittura."
    mount -n -o remount,rw /
    echo -n $>$/etc/mtab
    mount -f -o remount,rw /
    echo "Montaggio del filesystem locale."
    mount -a -t nonfs,nosmbfs
fi
#
# end of local\`-fs
```

Listing 3.3. /etc/init.d/local_fs

13. Ekzekutoni komandën më poshtë duke bërë të ekzekutueshem file e sapokrijuar
chmod +x local_fs

3.25 Startup

1. Krijoni file **menu.lst** në /boot/grub në hard diskun destinacion.
2. Populloni file me

```
default 0
timeout 3
color cyan/red white/blue
title MVux
root (hd0,0)
kernel /boot/vmlinuz root=/dev/sda1 rootdelay=10
boot
```

Listing 3.4. /boot/grub/menu.lst

3. Hyni në direktorinë /usr/src/sysvinit{versione}/src në usb
4. Ekzekutoni komandën **make**
5. cp init halt shutdown /mnt/hd/sbin
6. ln -s halt /mnt/hd/sbin/reboot
7. ln -s init /mnt/hd/sbin/telinit
8. mknod /mnt/hd/dev/initctl p
9. Krijoni file si më poshtë dhe ruheni me emrin /etc/**inittab** tek hard disku destinacion

```
# /etc/inittab - init daemon configuration file
#
# Default runlevel
id:1:initdefault:
#
# System initialization
si:S:sysinit:/etc/init.d/rc S
#
# Runlevel scripts
r0:0:wait:/etc/init.d/rc 0
r1:1:respawn:/bin/sh
r2:2:wait:/etc/init.d/rc 2
r3:3:wait:/etc/init.d/rc 3
r4:4:wait:/etc/init.d/rc 4
r5:5:wait:/etc/init.d/rc 5
r6:6:wait:/etc/init.d/rc 6
#
# end of /etc/inittab
```

Listing 3.5. /etc/inittab

10. Krijoni një file teksti si më poshtë dhe emërtojeni /etc/init.d/**rc**

```
#!/bin/sh
# /etc/init.d/rc - runlevel change script
#
PATH=/sbin:/bin
SCRIPT_DIR="/etc/rc$1.d"
#
# Check that the rcN.d directory really exists.
if [ -d $SCRIPT_DIR ]; then
# Execute the kill scripts first.
for SCRIPT in $SCRIPT_DIR/K*; do
    if [ -x $SCRIPT ]; then
        $SCRIPT stop;
    fi;
done;
# Do the Start scripts last.
for SCRIPT in $SCRIPT_DIR/S*; do
    if [ -x $SCRIPT ]; then
        $SCRIPT start;
    fi;
done;
fi
# end of /etc/init.d/rc
```

Listing 3.6. /etc/init.d/rc

11. chmod +x /mnt/hd/etc/init.d/rc

12. Modifiko **local_fs** duke shtuar seksionin për unmount të filesystem siç referohet më poshtë

```
#!/bin/sh
#
# local_fs - check and mount local filesystems
```

```

#  

PATH=/sbin:/bin ; export PATH  

case $1 in  

start)  

echo "Checking_local_filesystem_integrity."  

fsck -ATCp  

if [ $? -gt 1 ]; then  

echo "Filesystem_errors_still_exist!_Manual_intervention_required."  

/bin/sh  

else  

echo "Remounting/_as/_read_write."  

mount -n -o remount,rw /  

echo -n > /etc/mtab  

mount -f -o remount,rw /  

echo "Mounting_local_filesystems."  

mount -a -t nonfs,smbfs  

fi  

;;  

stop)  

echo "Unmounting_local_filesystems."  

umount -a -r  

;;  

*)  

echo "usage: _$0_start|stop";  

;;  

esac  

#  

# end of local_fs

```

Listing 3.7. /etc/init.d/local_fs

13. Krijoni file teksti si më poshtë dhe ruheni në /etc/init.d/**hostname**

```

#!/bin/sh  

#  

# hostname - set the system name to the name stored in /etc/hostname  

#  

PATH=/sbin:/bin:/usr/bin:/usr/sbin ; export PATH  

echo "Setting_hostname."  

if [ -f /etc/hostname ]; then  

hostname $(cat /etc/hostname)

```

```
else
    hostname mvux-linux
fi
#
# end of hostname
```

Listing 3.8. /etc/init.d/hostname

14. Krijoni file teksti dhe ruheni në /etc/init.d/**halt**

```
#!/bin/sh
#
# halt - halt the system
#
PATH=/sbin:/bin ; export PATH
echo "Initiating_system_halt."
halt
#
# end of /etc/init.d/halt
```

Listing 3.9. /etc/init.d/halt

15. Krijoni file të tipit tekst dhe ruheni në /etc/init.d/**reboot** në hard disk destinacion

```
#!/bin/sh
#
# reboot - reboot the system
#
PATH=/sbin:/bin ; export PATH
echo "Initiating_system_reboot."
reboot
#
# end of /etc/init.d/reboot
```

Listing 3.10. /etc/init.d/reboot

16. chmod +x /mnt/hd/etc/init.d/*

17. cd /mnt/hd/etc
18. mkdir rc0.d rc1.d rc2.d rc3.d rc4.d rc5.d rc6.d rcS.d
19. cd rcS.d
20. ln -s ../init.d/local_fs S20local_fs
21. ln -s ../init.d/hostname S30hostname
22. cd ../rc0.d
23. ln -s ../init.d/local_fs K10local_fs
24. ln -s ../init.d/halt K90halt
25. cd ../rc6.d
26. ln -s ../init.d/local_fs K10local_fs
27. ln -s ../init.d/reboot K90reboot

3.26 FAQ - Boot

- Device

Ne nisje para se tē hapet shell shfaqet një mesazh gabimi

```
VFS: Mounted root (ext2 filesystem) readonly.  
Freeing unused kernel memory: 332k freed  
INIT: version 2.86 booting  
Checking local filesystem integrity.  
ext2fs_check_if_mount: No such file or directory while determining whether /dev/sdb2 is mounted.  
fsck.ext2: No such file or directory while trying to open /dev/sdb2  
/dev/sdb2:  
The superblock could not be read or does not describe a correct ext2 filesystem.  
If the device is valid and it really contains an ext2 filesystem (and not swap  
or nfs or something else), then the superblock is corrupt, and you might try
```

```
running e2fsck with an alternate superblock:  
e2fsck b 8193 <device>  
Filesystem errors still exist! Manual intervention required.
```

Zgjidhje të mundshme

- "unable to open an initial console."
"vfs mounter root (ext2 filesystem), freeing unused kernel memory,
kernel panic: not syncing attempt to kill init"

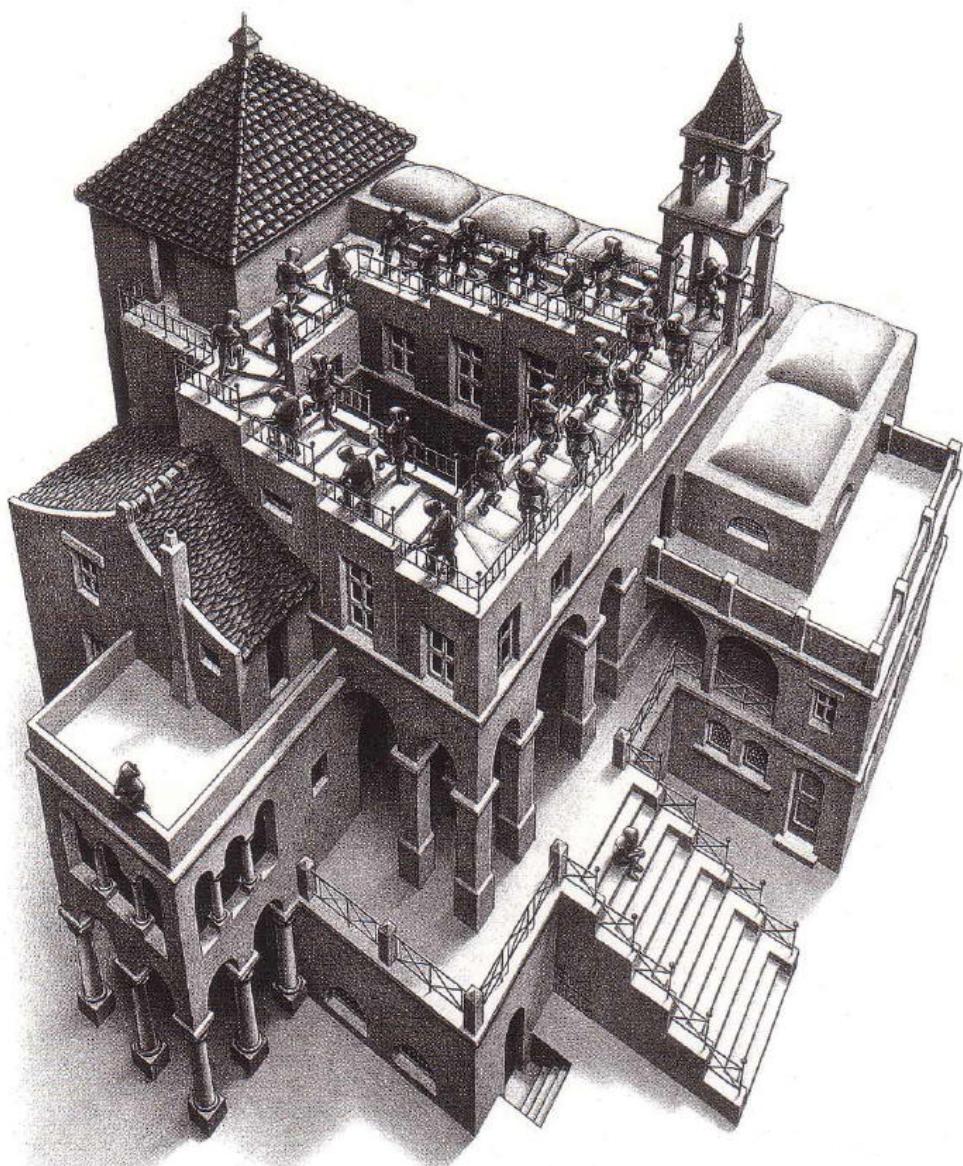
Operacionet në file suportohen nga Coreutils nën formen e shkruar, urdheruar, krahasuar, formatimit alternativ, vizualizimit dhe manipulimit alternativ të informacionit relativ të file. Ne nivelin e filesystem Coreutils ofron instrumenta baze për të drejtuar strukturen e filesystem duke e spostuar nga një nivel në një tjeter,kopjuar,fshire dhe riemëruar file dhe direktori të tera. Drejtimi i përdoruesve dhe i grupeve të përdoruesve identifikimi dhe informacioni i përdoruesve lejet për përdorim të file dhe perkufizimi i hierarkive të direktorive jane funksionalitetetë tjera të kesaj pakete. Programe të tjere të te njëjtët grup utilitetesh lejojnë krijimin e një particioni tjeter të tipit ext2,

Ne per gjithesi e2fsprogs i përdor utilitetet e tij për filesystem ext2 dhe me instalimin e tij në filesystem mund të menaxhohet më mire pëforma e një filesystem ext2. Ne pakete jane dhe disa librari të përdorura nga programe të tjere për të aksesuar direkt strukturat e filesystem ext2 në disk. Paketa e programeve e2fsprogs përfshin dhe një suport për filesystem të tipit ext3. menaxhimin e file system, console, particione dhe mesazhe dhe me menaxhimin e orës së sistemit

Opsionet:

Këtë radhë është e nevojshme të përcaktohen 2 linke simbolike të programeve të sapokopjuar që kanë një rëndësi të madhe duke qenë përgjegjës për fazën e fikjes së kompjuterit. Për funksionimin korrekt të *shutdown* është e nevojshme të krijosh një *pipe* me komandën *mknod /dev/initctl p*.

4 Ambienti Grafik



4.1 Server X

Kompilimi dhe instalimi i serverit X pa përdorur distribucionin pritës, është një test optimal për sistemin tonë. Menaxheri grafik standard i ambientit Unix është implementuar për të mundësuar përbërësit bazë të ndërfaqes grafike, menaxhimin e dritareve, tastierës, mouse. Interesi i veçantë për serverin X lidhet me rëndësinë e tij, por edhe me vështirësinë e procedurës së kompilimit. Vështirësia vjen sepse serveri X ka varësi nga shumë programe te tjera, që kërkohen në fazë konfigurimi. Kompilimi dhe instalimi korrekt i serverit X na ofron një ambient user-friendly, dhe na ndihmon të konstatojmë se kemi krijuar një sistem të qëndrueshëm dhe të personalizueshëm 100%.

4.1.1 Pse?

Duke qenë se nuk kemi instaluar akoma rrjetin, do i referohemi gjithmonë distribucionit pritës. Shkarkojmë nga rrjeti kodet për serverin X duke bërë kujdes që të marrim paketën e plotë, duke përfshirë edhe utility si xterm (emulatori i terminalit), xclock, xcal (utility me pak të rëndësishme, por që do jenë të nevojshme në fazën e testit). Dekomprimojmë arkivin dhe e kopojmë në hard diskun e jashtëm (pendrive). Ristartojmë distribucionin tonë dhe pozicionohemi te arkivi që shkarkuam. Procedura e kompilimit është ajo e zakonshmja. Fillimisht do hasni probleme për shkak të mungesës së librarive të nevojshme për një kompilim korrekt. Në këtë rast, mund të përdorni, për të mos humbur kohë, menaxherin e paketave për instalimin e programeve që ju nevojiten, duke mos harruar paketat e librarive me prapashtesë -dev. Një zgjidhje alternative është të instalohen paketat duke u nisur nga kodet. Në fazën e instalimit krijohen shumë arkiva dhe kopjohen shumë files në pozicione të ndryshme. Një nga file më të rëndësishëm gjendet në /etc/X11, dhe quhet *xorg.conf*.

Ky file përban konfigurimin e sistemit në lidhje me skedën grafike, pajisjet e input/output si tastiera, mouse dhe ekranit. Për cdo pajisje do specifikohen driverat dhe opsonet, si për shembull resolucioni i ekranit, emulimi i mouse me tre butona, etj.

Në fund të procesit të instalimit është e nevojshme të konfigurohet serveri X për pjesën e lidhur me programet që startohen. Kur ekzekutojmë komandën startx, startojmë serverin X. Për të ekzekutuar ndonjë program tjeter gjate startimit të sistemit duhet të modifikojmë file *.xinitrc*. Pra, nga ky file mund të rendisim programet që startohen gjatë nisjes së sistemit, duke përfshirë *window manager*. Një shembull shumë i thjeshtë për të testuar instalimin e serverit X është:

```
% fillimi file .xinitrc
```

```
xterm
```

```
% fundi file .xinitrc
```

Në këtë mënyrë kur startohet serveri X do të shfaqet edhe emulatori i shell. Mund të vini re mungesën e kornizave dhe të elementeve dekorativë të dritares. Për të startuar programe të tjera mjafton të rendisni ato duke i ndarë me një & si në shembullin më poshtë:

```
% fillimi file .xinitrc
```

```
xterm &
```

```
xclock &
```

```
xcalc
```

```
% fundi file .xinitrc
```

Për të mbyllur serverin X mund të përdorni tastet (Ctrl + Alt + Backspace) ose të mbyllni programin e fundit të hapur, në shembullin më sipër ai do ishte makina llogaritëse. Duke qenë se nuk kemi akoma një window manager, është më komode të vendosim si program të fundit në *.xinitrc* programin xterm, me të cilin është më i thjeshtë bashkëveprimi.

4.2 Window Manager

4.2.1 Pse?

Window manager është shumë i rëndësishëm për bashkëveprimin me përdoruesin. Dy ambientet më të përdorur janë Gnome dhe KDE. Këto janë shumë user-friendly, por janë shumë të ngarkuara përsa i përket hapësirës në hard disk. shtë e këshillueshme të përdoret një window manager që është user-friendly, por më të thjeshtë në instalim dhe përdorim. Instalimi i tyre konsiston në instalimin e një pakete të vetme, në ndryshim nga ambientet që përmendëm më sipër. Disa shembuj janë:

- fluxbox
- icewm
- evilwm

4.2.2 How to

Instalimi i këtyre window manager thjeshtohet nga programet e menaxhimit të pake-tave. Thjeshtësia e tyre vjen sepse zakonisht ato instalohen me një paketë të vetme, në një kohë që KDE ka nevojë për shumë paketa të lidhura midis tyre. Pas instalimit procedura për startimin është ajo që thamë më sipër. Për shembull, për të startuar ambientin **fluxbox** është e nevojshme të editojmë file `.xinitrc` si më poshtë:

```
fluxbox          &
xclock -geometry +0-0  &
xterm
```

Sipas nevojës mund të specifikoni edhe opsione për hapjen e programeve duke përcaktuar edhe pozicionin në ekran për të evituar mbivendosje. Nga ky moment mund të përdorni dritatët si objekte të modifikueshme në pozicion dhe përmasa.

Duke klikuar me tastin e djathtë të mouse në desktop keni në dispozicion një menu kontekstuale plotësisht të personalizueshme. Në fakt, menuja përcaktohet plotësisht nga një file teksti që fillimisht e gjejmë në `/etc/X11/fluxbox/system.fluxbox-menu`. Për ta përdorur mjafton të kopjohet në `/root/.fluxbox` e të emërtohet *menu*

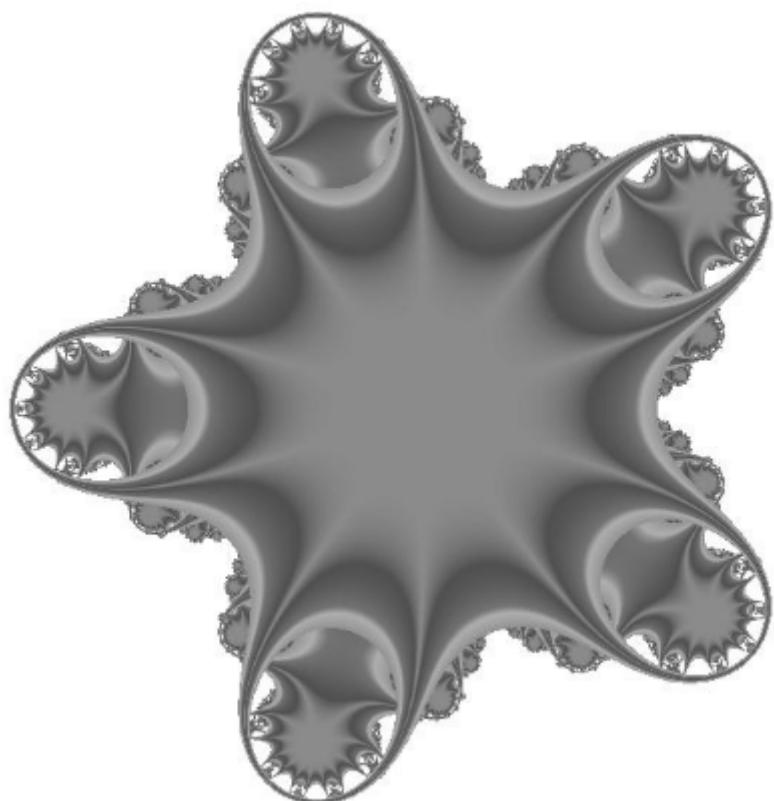
```
# This is an automatically generated file.  
# Please see <file:/usr/share/doc/menu/README> for information.  
  
# to use your own menu, copy this to ~/.fluxbox/menu, then edit  
# ~/.fluxbox/init and change the session.menuFile path to ~/.fluxbox/menu  
  
[begin] (Fluxbox)  
include-menu-defs  
[config] (Configuration)  
[submenu] (Styles) {}  
[stylesdir] (/usr/share/fluxbox/styles)  
[stylesdir] (~/.fluxbox/styles)  
[end]  
[workspaces] (Workspaces)  
[reconfig] (Reconfigure)  
[restart] (Restart)  
[exit] (Exit)  
[end]
```

Ky file mund të personalizohet sipas dëshirës. Vini re praninë e një *submenu* ku mund të shtohen edhe programe të tjera. Për shembull kodi i mëposhtëm

```
[submenu] (Programe){}  
[exec] (FireFox) {firefox}
```

```
[exec] (Kedit) {Kedit}  
[end]  
[submenu] (Te tjera) {}  
    [exec] (Ora) {xclock}  
    [exec] (Makina llogaritese) {xcalc}  
[end]
```

jep si rezultat dy menu që quhen Programe dhe Të tjera, brenda të cilave kemi lidhjet me programet e listës. Vini re se përbajtja e kllapave () është ajo qe shohim kurse përbajtja e {} është komanda që ekzekutohet kur klikojmë aty.



5.1 Si?

5.1.1 DPKG

1. Shkarko nga interneti kodin e dpkg në faqen
http://ftp.debian.org/debian/pool/main/d/dpkg/dpkg_1.14.4.tar.gz
2. Dekomprimo arkivin e shkarkuar në **/usr/src**
3. Ekzekuto komandën **./configure**
4. Ekzekuto komandën **make**
5. Shko në arkivin src dhe kopjo dpkg në arkivin **/usr/bin** të hard diskut të jashtëm
6. Ekzekuto komandën **cd ../dpkg-deb** dhe kopjo **dpkg-deb** në **/usr/bin** të hard diskut të jashtëm
7. Kopjo skriptet që janë në **script** duke i hedhur në **/usr/bin** të hard diskut të jashtëm

```
apt-get install -d -y --reinstall  
Lista e paketave për të instaluar apt
```

- apt
- libc6
- libgcc1
- libstdc++6
- debian-archive-keyring
- gcc-4.2-base
- gnupg

- libbz2-1.0
- libldap2
- libreadline5
- libusb-0.1-4
- zlib1g
- gpgv
- makedev
- base-passwd
- readline-common
- libncurses5
- libgnutls13
- libsasl2-2
- libdb4.4
- libgcrypt11
- libgpg-error0

krijo arkivin /etc/source.lst

5.1.2 Gcc

1. Hap një dritare shell
2. Edito arkivin /etc/sources.list në

```
deb http://mi.mirror.garr.it/mirrors/debian/ lenny main
deb-src http://mi.mirror.garr.it/mirrors/debian/ lenny main
deb http://security.debian.org/ lenny/updates main contrib
deb-src http://security.debian.org/ lenny/updates main contrib
```

3. Shko në arkivin `/var/cache/apt/archive`
4. Ekzekuto komandën `rm *.deb`
5. Hap programin `synaptic`
6. Seleksiono paketat për instalim sipas listës
 - `libc6`
 - `libgcc1`
 - `libgomp1`
 - `gcc`
 - `gcc-4.2`
 - `cpp`
 - `cpp-4.2`
 - `gcc-4.2-base`
 - `binutils`
 - `libc6-dev`

7. Shko në arkivin `/var/cache/apt/archives`
8. Krijo një arkiv teksti dhe quaje `scompattadeb`

```
#!/bin/sh
for i in *.deb;
{
    dpkg-deb -x $i $1
}
```

9. Ndrysho të drejtat e ekzekutimit të arkivit të krijuar me komandën `chmod +x scompattadeb`

-
10. Ekzekuto skriptin e krijuar duke përdorur komandën `./scompattadeb gcc`
 11. Verifiko se ekzekutimi përfundoi me sukses dhe arkivi gcc është plotesuar
 12. Kopjo përmbajtjen e arkivit gcc në hard diskun e jashtëm.

5.2 Pse?

5.2.1 Dpkg

Menyra me e thjeshte për instalimin e pjeses me të madhe të programeve në Linux është duke përdorur paketat. Paketat administrohen nga disa programe, të cilat befne të mundur instalimin, heqjen dhe modifikimin e tyre. Ekzistojne disa distro Linux dhe administrues të ndryshëm për paketat përkatëse. Ky tekst përqendrohet në administrimin e paketave për distro Debian. Në Debian Linux ka shumë programe për administrimin e paketave, por të gjitha bazohen në programin dpkg. Dpkg (Debian PacKaGe) është programi themelor për sistemin e administrimit të paketave Debian. Bëhet fjalë për një program që mund të ekzekutohet nga rreshti i komandave dhe ka opsione të ndryshme si `-i` për instalimin ose `-r` për heqjen e paketave në input. Per momentin këto opsione janë të mjaftueshme. Karakteristika e këtij programi është se, në versionin bazë, administruesi nuk kujdeset për të kontrolluar dipendencat. Ky opzion, që është i nevojshëm, është i parashikuar në programet si apt-get, aptitude, synaptic që në modalitetet grafike ose tekstuale lejojnë instalimin e një pakete dhe të paketave të lidhura me të . Mungesa e kontrollit kërkon më shumë kujdes gjatë fazës së instalimit në rast gabimesh të mundshme.

5.2.2 Gcc

Instalimi i kompilatorit të gjuhes C është një pjesë shumë e rëndësishme dhe kërkon vëmendje të veçantë. Në këtë rast nuk do aplikohet procedura normale për kompilim dhe instalim sepse kompilimi kërkon vetë kompilatorin C.

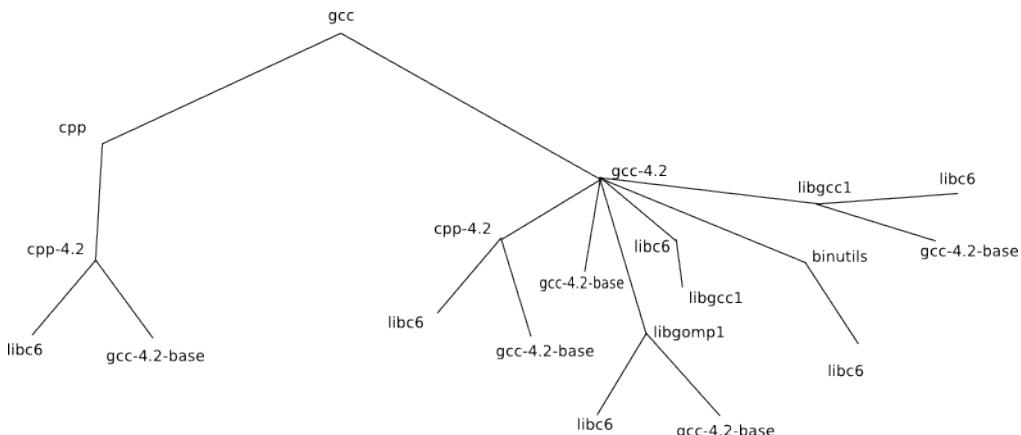


Figura 5.1. Pema e dipendencave gcc

Me sipër shikoni listën e paketave që nevojiten për një instalim korrekt të kompilatorit gcc version 4.2. Për realizimin e projektit është zgjedhur të instalohet gcc me paketat për Debian. Këto paketa janë arkiva të komprimuar që kanë brenda tyre dy file data.tar.gz dhe control.tar.gz që përfaqësojnë reperktivisht arkivat e kompiluar dhe të organizuar në arkiva të filesystem (gati për përdorim) dhe disa file për kontrollin e respektimit të dipendencave për ekzekutimin e programit. Në run-time paketat do të instalohen me programe grafike më të thjeshta për përdorim dhe konfigurim. Për këtë fazë u përdor komanda `dpkg-deb` që midis të tjerash shpërbën file në input dhe krijon një arkiv në output në të cilin do dekomprimohen arkivat në data.tar.gz. Duke përdorur një lidhje në rrjet ose me një tool grafik në Debian, siç është synaptic, mund të shkarkohen files .deb që janë në arkivin default të programit është /var/cache/apt/archives. Për të thjeshtuar gjërat mund të boshatiset arkivi default me komandën `rm *.deb`, shkarkohen me synaptic paketat e nevojshme, krijohet një

skript brenda arkivit për të automatizuar administrimin e files:

```
#!/bin/sh

for i in *.deb;
{
    dpkg-deb -x $i gcc
}
```

Në vijim:

1. ruhet ky file me emrin *scompattadeb*
2. i jepen të drejta ekzekutimi me komandën `chmod +x scompattadeb`
3. ekzekutohet skripti i krijuar duke përdorur komandën `./scompattadeb`

Rezultati është krijimi i një arkivi `gcc` që përmban një sërë arkivash që duhet të kopjohen në hard diskun e jashtëm.

5.3 Programë të domosdoshme

Në këto faqe kemi parë si krijohet një sistem Linux minimal që startohet nga një hard disk i jashtëm. Tani sistemi operativ mund të kryejë veprime të thjeshta si login dhe krijim arkivash duke përdorur komanda shell minimale. Per të bere këtë na u desh të kemi një sistem pritës (host) që shërben si "placente" për distribucionin që krijuam. Për të shkëputur përfundimisht sistemin e krijuar na duhet të instalojmë disa programe dhe disa librari zhvillimi që dojenë baza e sistemit të ri dhe bëjnë të mundur kompilimin e programeve direkt në distribucionin tonë. Për të bere të mundur instalimin e paketave, ato duhet të jene në hard diskun tonë. Alternativat janë të shumta: `cd/dvd`, hard disk i jashtëm, internet. Duke qenë se rrjeti nuk është instaluar akoma, zgjidhja më e shpejtë është duke përdorur edhe një here të fundit distribucionin host. Në fakt, me komandën `mount` mund të krijohet një pike

montimi ku të gjejmë gjithë particionin e distribucionit host. Brenda këtij arkivi mund të kalohet në arkivin `/var/cache/apt/archives` ku ndodhen zakonisht të gjitha paketat e instaluar. Siç thamë, për të instaluar një paketë është e nevojshme që atë ta kemi në kompjuterin tonë. Prandaj të gjitha programet që administrojnë paketat përdorin atë arkiv si depozitë lokale. Për qëllimet tona çdo paketë duhet instaluar nëpërmjet komandës `dpkg -i emripaketes.deb`. Fillimisht duhet të instalojme një kompilator për gjuhen C, në këtë rast gcc. Siç e pamë, procesi i komplilit në ambient Linux paraprihet nga verifikimi i konfigurimit të sistemit. Në mënyrë të veçante skripti i konfigurimit verifikon nëse kemi në sistem të gjithë kompilatorët, libraritë, programet e nevojshme për komplimin dhe për funksionimin korrekt të programit që jemi duke instaluar. Eshtë e kuptueshme se nëse instalohet vetëm paketa gcc, nuk do jetë e mjaftueshme për të pasur një sistem të qëndrueshëm dhe të gatshëm për komplimin e programeve të tjera.

Këshillë: Eshtë mire të testohet sistemi me një program C që stampon në video një mesazh të thjeshtë teksti.

5.4 Praktikë

5.4.1 Probleme

GCC (GNU Compiler Collection, fillimisht GNU C Compiler) është një set kompilatorësh të krijuar fillimisht nga themeluesi i Free Software Foundation Richard Stallman, që dëshironë një kompilator që të ishte software i lirë (free software). GCC ka si qëllim parësor përkthimin e kodit në gjuhën e makinës në mënyrë që të ekzekutohet. Për shëmbull, për të kompiluar një file me kod të shkruar në C, mund të ekzekutohet gcc duke i dhënë në input këtë file. Për të administruar aspektet e ndryshme dhe detajin e komplilit mund të jepen një numer shumë i madh parametash; i vetmi parametër i detyrueshëm është emri i file që do kompilohet. Një

shëmbull elementar kompilimi është komanda:

```
gcc main.c
```

që urdhëron kompilatorin të lexojë main.c (.c është zgjatimi karakteristik për files që kanë kod të shkruar në C) dhe ta kompilojë. Arkivi i ekzekutueshëm quhet a.out sepse nuk është specifikuar ndonje emër tjetër për file në output. Kjo do të bëhej duke përdorur parametrin -o. Kompilimi kalon nëpër tre faza:

1. file që kompilohet trajtohet nga preprocesori, i cili analizon direktivat e shkruara për kompilatorin (zakonisht këto ndodhen në fillim të kodit dhe kanë përpara shenjen #). Nëse gjen parametrin -E, gcc ekzekuton vetëm këtë hap duke dhënë në output rezultatin e aktivitetit të preprocesorit.
2. në fazën e dytë krijohet një file objekt, që nuk është akoma i ekzekutueshëm
3. në këtë pike hyn në funksion linkeri që bashkon të gjitha file objekt dhe librarite duke prodhuar një file gjithëperfshires.

Nëse do konsideronim më në detaj instalimin, mund të përdoret procedura e mëposhtme duke filluar nga përdoruesi root. Duke lëvizur te arkiva në të cilën kemi dpkg, domethënë në /var/lib/apt/archives ekzekutohen komandat:

```
dpkg -i binutils*.deb
```

```
dpkg -i gcc*.deb
```

```
dpkg -i cpp*.deb
```

```
dpkg -i libc6-dev*.deb
```

Këto komanda shërbejnë për të instaluar të gjitha paketat e nevojshme për funksionimin e gcc.

Ndoshta duhet të dini se shpesh, kur zgjidhni të instaloni një pakete të caktuar, programet e instalimit vendosin automatikisht instalimin e disa paketave të tjera që janë të nevojshme për mirëfunksionimin e tyre. Më poshtë renditen karakteristikat

e GCC që e bëjnë të dallohet nga kompilatorët e tjerë:

- Portabilitet i lartë. GCC në ditët e sotme ekzekutohet në një numer të madh sistemesh operative dhe në platforma të ndryshme.
- Standard. Zhvilluesit e GCC janë munduar që të respektojnë të gjitha standardet e gjuhës për të siguruar një kompatibilitet të lartë. Në veçanti, gcc ka parashikuar përdorimin e parametrit ”-ansi” që siguron kompatibilitetin total me standardin ANSI dhe mënjanon të gjitha specifikat e platformës se përdorur.
- Optimizim. Nga të gjithë kompilatorët e testuar, GCC është ai me optimizuesin më të sofistikuar. Për platformat x86, GCC parashikon parametra optimizimi të kodit për pjesën më të madhe të procesorëve.
- Opsione. Nga të gjithë kompilatorët, GCC është ai me numrin më të madh të parametrave që krijojnë mundësinë për të pasur kontroll të plotë mbi kodin që gjenerohet, mbi dialektet e gjuhëve, mbi ’warning’ dhe mbi mënyrën sesi procesi përfundon.
- Open source. GCC ishte kompilatori i parë C/C++ i lirë. Mund të merret kodi i programit dhe të shikohet se si funksionon. Dokumentimi i kodit është shumë cilësor dhe mund të korrigohen gabimet nëse ka.

5.4.2 Disa vlerësime

Në këtë pjese është përmendur rëndësia e një prej problemeve që hasen më shpesh gjatë ndërtimit të një distribucioni. Eshtë rasti i mungesës në sistem të një ose më shumë librarive që janë të nevojshme për komplimin e një programi të caktuar. Eshtë e vështirë që të jepen udhëzime të sakta për zgjidhjen e këtyre problemeve

sepse ato ndryshojnë nga kompjuteri në kompjuter, por është e rëndësishme që të merren parasysh nga ata që i futen kësaj rruge. Ndodh shpesh gjatë instalimit të disa paketave që në një distribucion mungojnë disa librari të caktuara dhe në një tjetër distribucion mungojnë librari të tjera. Shkaku gjendet zakonisht në mënyrën e instalimit të distribucionit ose nga një instalim paketash që nuk ka përfunduar me sukses. Për të evitar këtë problem, që haset edhe në procesin e ndërtimit të MVux, duhet ruajtur një konfigurim optimal dhe duhen shfrytëzuar njohuritë e secilit për të kapur sa më shpejt mesazhet e gabimit në mënyrë që të korrigohen në kohë.

5.5 Test

1. Ristarto kompjuterin me usb hard disk
2. Nëqoftëse çdo gjë ka shkuar mirë, duhet të na shfaqet ambienti grub
3. Ekzekuto komandën `root(hd0,0)`
4. Ekzekuto komandën `kernel /boot/vmlinuz root=/dev/sdb1 rootdelay=10 init=/bin/bash`
5. Ekzekuto komandën `boot`
6. Kur dalim në shell testojmë programet e sapokopjuara nëpërmjet komandave `ls`, `date`, `df`.
7. Verifikojmë se nuk është e mundur të ekzekutohen disa komanda si `mkdir`, `cp`, `mv` sepse filesystem është akoma në modalitet `read-only`
 - Ristarto kompjuterin
 - Vendos hard diskun e jashtëm

- Futu në ambientin e grub

- Ekzekuto komandën *root(hd0,0)*

- Ekzekuto komandën

```
kernel /boot/vmlinuz root=/dev/sdb1 init=/bin/bash rootdelay=15 ro
```

- Përcakto ndryshoren e ambientit PATH nëpërmjet

```
PATH=/sbin:/bin:/etc/init.d ; export PATH
```

Tani mund të ekzekutohen të gjithë skriptet dhe programet që janë në ato pozicione.

- Verifikojmë përbajtjen (bosh) të *mtab* me komandën

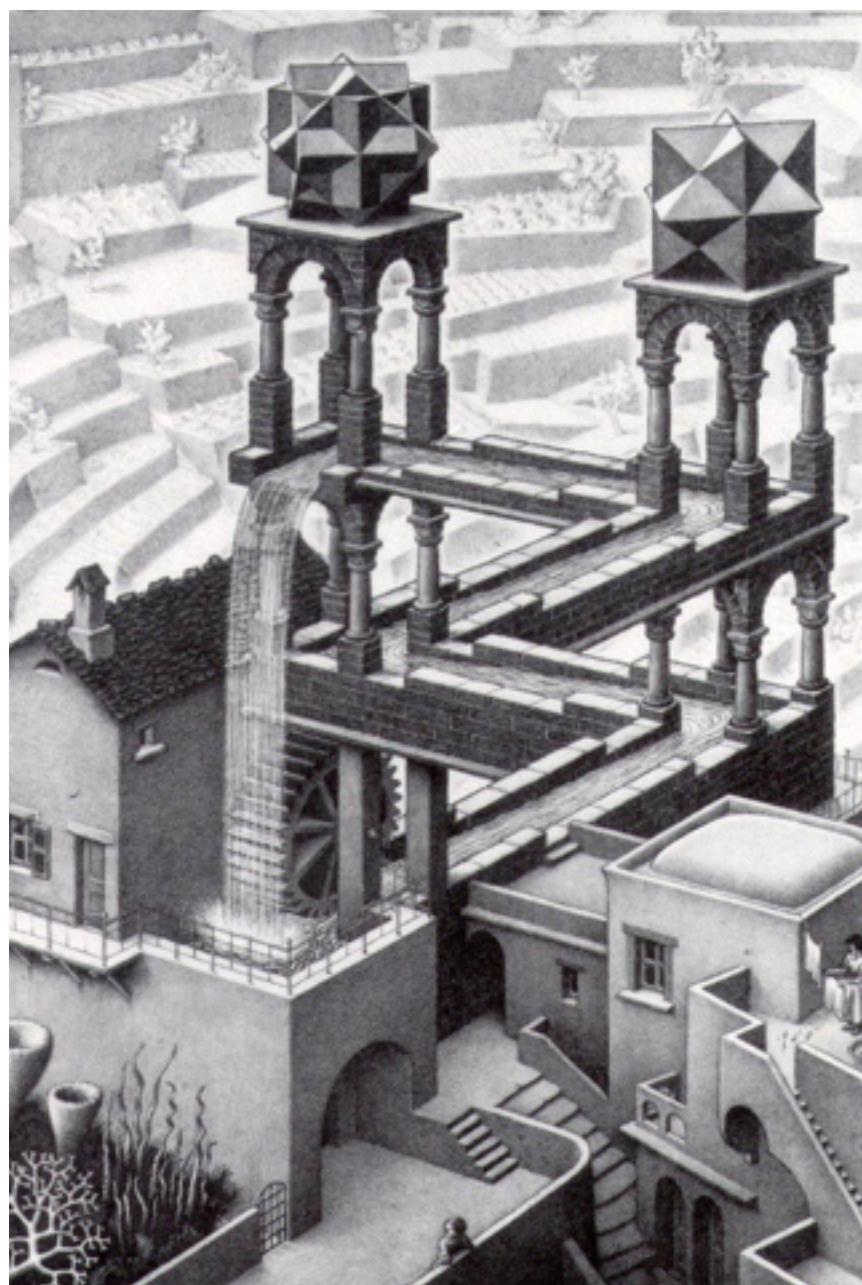
```
cat /etc/mtab
```

- Ekzekutojmë komandën *local_fs*

- Verifikojmë se *mtab* nuk është me bosh.

- Testojmë komandën *df* për të marrë informacion mbi pajisjet e montuara.

6 Rrjeti



6.1 Si?

6.1.1 Kompilimi i moduleve

1. Verifiko nëse ekziston një pajisje rrjeti me komandën **lspci**
2. Shëno markën dhe modelin e skedës se rrjetit;
3. Konfiguro si driver në kernel në mënyrë që të përfshihet me pas me komandat **insmod** ose **modprobe**
4. Konfiguro kernel me komandat me poshte:
5. **make modules; make modules_install**
6. Me **make modules_install** krijohet automatikisht arkivi në **/lib/modules**
7. Kopjo arkivin e krijuar në hard diskun e jashtëm duke respektuar PATH

6.1.2 Net-Tools

- Për të bërë të mundur përdorimin e rrjetit duhet shkarkuar versioni i fundit i paketës Net-tools nga faqja
<http://www.tazenda.demon.co.uk/phillip/net-tools/net-tools-1.60.tar.bz2>
- Dekomprimo arkivin në **/usr/src**
- Shko në **/usr/src/ net-tools-1.60.tar**
- Ekzekuto komandën nga shell **make config**
- Ekzekuto komandën **make**
- Ekzekuto komandën **make install**

- Shko në nenarkivin src
- Aty gjen kodin e komandave për veprimet bazë në rrjet, kodin objekt pas kompilimit dhe arkivin e ekzekutueshem të sapokrijuar
- Kopjo komandat me të rëndësishme dhe me të zakonshme nga ky arkiv në arkivin bin të hard diskut të jashtëm duke përdorur komandën cp. Për shembull me: `cp hostname ifconfig route /mnt/hd/bin`
- Verifiko nëse duhen librari të tjera nëpërmjet komandës ldd sic është bërë për paketat e tjera
- Kopjo librarite e nevojshme duke respektuar PATH;
- Ekzekuto dhclient për të vënë në funksion rrjetin;
- Ekzekuto ping {adresa IP e destinacionit} për të verifikuar se rrjeti funksionon.

6.2 Si?

Nevoja për të krijuar një rrjet për përdorimin e përbashket të burimeve, ndermjet dy ose me shume kompjuterave, është një nga kërkesat me të medha të shoqerise informatike. Te realizosh një rrjet lokal (LAN, Local Area Network) do të thotë të lidhesh dy ose me shume PC me njëri tjeterin. Potenciali i këtij veprimi është i madh: për shembull, brenda rrjetit cdo kompjuter mund të ”shikoje” informacionet në kompjuterat e tjerë, duke bërë të mundur shkëmbimin e të dhënavë, përdorimin e përbashket të pajisjeve, etj. Pra, objktivi është të realizohet komunikimi i dy hard disqeve që jane të lidhur në dy kompjutera të ndryshem dhe për të bërë këtë duhet pare nëse ne këto kompjutera jane të instaluar pajisjet e rrjetit. Në shell ekzekutohet komanda `lspci`, që liston të gjitha pajisjet e lidhura me kompjuterin.

Pasi ekzekutohet komanda na jep informacionin e duhur. Në rastin tone pajisja e rrjetit është: *00:0a.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+ (rev 10)*. Duke patur parasysh këto të dhëna mund të rikonfigurohet kernel për të suportuar këtë pajisje. Pasi gjendet në kernel skeda e rrjetit, kompilohet ai përbërës si modul i jashtëm. Me rikonfigurimin e kernel mund të përsëritet procedura për komplimin, sic e permendem me siper. Për të kompiluar edhe modulet, duke përfshire keshtu skedën e rrjetit, ekzekutojmë komandat **make modules** dhe **make modules_install** për instalimin e moduleve në arkivin /lib/modules/{versione kernel}, duke krijuar edhe arkivat që përbajne dipendentcat midis moduleve të ndryshme. Mund të ekzekutohen të gjitha këto komanda se bashku duke përdorur 'make bzImage && make modules && make modules_install', në mënyrë që procedura të vazhdojë në mënyrë automatike njera pas tjetres. Në këtë mënyrë krijohet një arkiv brenda /lib/modules që do kopjohet në hard diskun e jashtëm duke respektuar gjithmone PATH.

6.3 Test

6.3.1 Webserver e mini-chat

Web server është një program që ofron një faqe web pas një kërkese të programit browser (shpesh në HTML). Informacionet e dërguara nga web server udhetojnë në rrjet nëpërmjet protokollit. Bashkesia e web server që gjenden në Internet formojnë WWW ose World Wide Web, që është një nga sherbimet me të shfrytezuara në Rrjetin e Madh. Normalisht web server vendoset në sisteme të dedikuara, por mund të ekzekutohet në kompjutera që kane server të tjera ose që përdoren për qëllime të tjera. Për qëllimet e këtij libri, do realizohet një komunikim i thjeshtë në rrjet duke instaluar një web server dhe një chat minimal. Realizimi i këtij funksionaliteti do behet e mundur nëpërmjet socket, kanal komunikimi i nevojsphem për aplikacionet

që ekzekutohen në një host.

6.3.2 Socket

Socket është një kanal komunikimi që bën të mundur dërgimin dhe marrjen e të dhënavë nëpërmjet rrjetit duke u mbështetur në një protokoll komunikimi. Në fakt nga anglishtja socket perkthehet “prize” dhe në fakt mund të imagjinohet si një prize korrenti kurse fluksi i të dhënavë mund të imagjinohet si rryma që kalon në percjelles. Protokollet e përdorshme për implementimin e Socket janë:

- protokolli TCP (Transfer Control Protocol)
- protokolli UDP(User Datagram Protocol)

Të dy protokollet mbështëten në protokollin IP (Internet Protocol). Protokolli TCP është një sherbim i besueshem për dorëzimin e paketave, orientuar nga lidhja (Connection-Oriented Acknowledged Transport Protocol) Protokolli TCP është i besueshëm sepse: garanton dorëzimin e paketave, kontrollon integritetin e të dhënavë të transmetuara duke përdorur llogaritjet checksum[?], garanton sekuencën e saktë dhe dorëzimin sipas radhes të te dhënavë të segmentuara, dërgon mesazhe pozitive në rast se të dhënat u dorëzuan ose mesazhe negative për të dhënat e padorëzuara. Kjo eviton duplikatet gjate dërgimit të një segmenti ose ridërgimin në rast gabimesh në transferim.

Protokolli TCP është i orientuar nga lidhja (connection oriented) duke qene se realizohet një lidhje midis dy njesive host dhe të dhënat ndahen në shume segmente me numra që kalojnë në një bytes stream. Avantazhet kryesore të këtij protokolli vijnë nga mundesia për të transferuar në mënyrë të sigurte një sasi të madhe të dhënash dhe në rast mbingarkesë të linjes, ky protokoll garanton drejtimin korrekt të paketave. Dizavantazhet jane të lidhura me ngadalesine në transferimin e të dhënavë për shkak të kontolleve që behen për të siguruar një transmetim efektiv të

paketave dhe për permasat e segmenteve. Për me teper, suporti është ekskluziv për komunikimin point to point (PTP) drejt një marresi të vetem dhe jo i tipit point to Multipoint (PTM) drejt shume marresve. Disa shembuj protokolle të TCP që përdoren me shpesh jane: FTP - Telnet - http.

6.3.3 Ambienti i punes

Ambienti i punes me siper është një laborator me disa kompjutera të lidhur në një rrjet yll, i ashtuquajturi **LAN**. Rrjeti lokal administrohet në mënyrë të centralizuar, gje që i jep një nivel sigurie të larte, dhe përdor një router për lidhjen në internet. Perdorimi i një routeri në vend të një kompjuteri që sherben si Server/Gateway është një zgjidhje ideale nëse nuk disponojmë një kompjuter për këtë qëllim. Në fakt, një kompjuter që do sherbente si Server/Gateway, perveçvojes për një software proxy të veçante, do duhet të ishte gjithmone i ndezur (edhe kur nuk bente ndonje pune tjetër). Sidoqofte, edhe nëse përdorimi i kompjuterit si Gateway si stacion që justifikon gjendjen e ndezur gjate gjithe kohes, ajo mund ta ngadalesojë tejmase punen sepse ka mbingarkesë aktivitetesh. Nga ana tjetër, një router është pajisje qeka vetem këtë funksion, pra analizimin dhe dërgimin paketave në destinacionin e duhur. Si rrjedhim, routeri e kryen funksionin me mire dhe qendron i ndezur aq sa nevojitet. Pervekësaj, një router nuk ze shume vend, nuk është i zhurmshem dhe nuk prodhon nxehesi sa një kompjuter që qendron vazhdimit i ndezur.

Perpara se të shpjegojmë shkurtimisht tekniken e përdorur nga router, duhet të saktësojmë se kompjuterat klientë kane një adresë IP private, kurse router gateway ka një adresë IP private dhe një adresë IP publike. Për shembull, një kompjuter klient që do të navigojë dërgon tek router gateway, nëpërmjet browser, një kërkese për hapjen e një adresë web të caktuar. Ky i fundit ”shikohet” nga kompjuteri klient nëpërmjet adresës IP private. Kerkesa si dhe të gjitha informacionet që qarkullojnë në rrjet, transmetohen në forme paketash TCP/IP: këto shoqerohen nga adresa

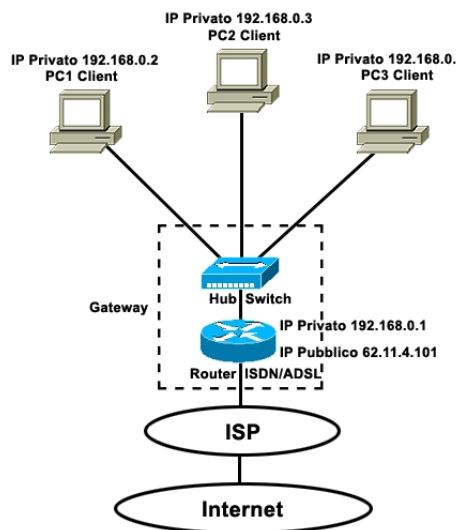


Figura 6.1.

IP e dërguesit dhe nga adresa IP e marresit (destinacionit). Router gateway merr kërkesen, por, para se ta dërgoje, zevendeson adresen IP private të kompjuterit klient me adresen IP publike të routerit (qe jepet nga ofruesi i sherbimit në momentin e lidhjes, p.sh.: 62.11.4.101). Në këtë mënyrë, serveri web që mban faqet shikon vetem adresen IP publike të gateway dhe jo adresen IP private të kompjuterit klient. Se fundi, në momentin kur router gateway merr informacionin nga interneti, i analizon dhe i transmeton tek kompjuteri klient që dërgoi kërkesen. Nese routeri është konfiguruar për të përdorur DHCP server, nuk është nevoja që të percaktojë adresën IP private statike: do jete DHCP që cakton adresën IP private në mënyrë dinamike sa here që kompjuteri ristartohet. Në këtë mënyrë konfigurimi është akoma me i thjështë. Nese është nevoja të shtohen kompjutera të tjera në rrjet, mjafton tu caktohet një adresë IP private progresive në lidhje me ato të vendosura nga router gateway.

6.3.4 Hyrja në rrjet nga pendrive

Vendosni pendrive në një nga kompjuterat e lidhur në rrjet. Pasi modifikohet bios, dhe pasi sistemi startohet nga pendrive, bejme ate që behet zakonisht për të verifikuar gjendjen e sistemit. Pra, ekzekutojmë një nga komandat për networking që ndodhen në net-Tools: ifconfig. Kjo komande krijon mundesine të percaktohen karakteristikat e nderfaqesve të rrjetit dhe nëse përdoret pa opsione na jep gjithe nderfaqesit aktive. Me pak fjale, listohen të gjitha vlerat e konfigurimit të rrjetit TCP/IP dhe përditeson të dhënët e protokollit DHCP (Dynamic Host Configuration Protocol) dhe sistemit DNS (Domain Name System). Nje shembull listimi mund të jete si me poshte:

```
eth0      Link encap:Ethernet HWaddr 00:01:02:2F:BC:40
          inet addr:194.177.127.234 Bcast:194.177.127.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:82075264 errors:0 dropped:0 overruns:0 frame:0
          TX packets:51585638 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:2858378779 (2.6 GiB) TX bytes:2524425895 (2.3 GiB)
          Interrupt:10 Base address:0x8800

eth1      Link encap:Ethernet HWaddr 00:E0:7D:81:9C:08
          inet addr:192.168.1 Bcast:192.168.1.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
          Interrupt:9 Base address:0x6000

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:10226970 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10226970 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1385547296 (1.2 GiB) TX bytes:1385547296 (1.2 GiB)
```

Pasi arritem në këtë pike, ekzekutojmë komandën dhclient për t'u lidhur me rrjetin lokal që përdor DHCP për të dhene adresat e rrjetit. Nese me pare jane bërë gabime në konfigurimin e kernel, për shembull nëse nuk jane përfshire modulet e skedës se rrjetit, si rezultat do kemi një mesazh gabimi gjate ekzekutimit të

dhclient dhe lidhja do të deshtoje. Nese cdo gje ka shkuar mire, do shfaqet mesazhi i meposhtem:

*There is already a pid file /var/run/dhclient.pid with pid 3571
killed old client process, removed PID file
Internet Systems Consortium DHCP Client V3.0.4
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/*

*Listening on LPF/eth0/
Sending on LPF/eth0/
Listening on LPF/eth1/00:18:f3:33:6b:26
Sending on LPF/eth1/00:18:f3:33:6b:26
Sending on Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 4
DHCPREQUEST on eth1 to 255.255.255.255 port 67
DHCPACK from 192.168.1.254
bound to 192.168.1.2 – renewal in 271066 seconds*

.

Pasi konfigurohen nderfaqesit, duhet verifikuar nëse rrjeti funksionon duke ”pinguar” një tjetër pajisje (kompjuter). Komanda ping sherben për të dërguar një pakete speciale që kërkon një pergjigje nga kompjuteri destinacion. Komanda ekzekutohet duke specifikuar adresen IP të kompjuterit ”objektiv”, në rastin tone 192.168.1.2. Nese rrjeti funksionon, atehere mesazhi që do shfaqet do jete i ngashhem me mesazhin me poshte:

*PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.034 ms*

```
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.033 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.033 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.033 ms
64 bytes from 192.168.1.2: icmp_seq=5 ttl=64 time=0.036 ms
64 bytes from 192.168.1.2: icmp_seq=6 ttl=64 time=0.036 ms
64 bytes from 192.168.1.2: icmp_seq=7 ttl=64 time=0.036 ms
```

Komanda dërgon një pakete cdo sekonde dhe, në rast se merr gjithmone pergjigje, raporton kohen e nevojshme. Kur nderpritet ekzekutimi me Ctrl-C, shfaqet edhe një statistike. Kjo na thotë se kompjuteri me adresë IP 192.168.1.2 është aktive dhe e arritshme. Pasi marrim këtë rezultat, pra dy disqe të jashtëm që "komunikojne" me njëri tjetrin nga dy vende të ndryshme, tanime mund të instalohet një web server, për të cilin thame diçka me siper, që i vendos në komunikim real dy disqet e jashtme nëpërmjet një mini-chat.

6.3.5 Realizimi i web server

Duke shfrytezuar funksionet e rrjetit që shtuam në distribucionin e realizuar, u mendua të implementohej një web server shume i thjeshtë, shkruar në gjuhen C. Ky program sherben si server dhe mund të sherbeje disa clients të pavarur nga njëri tjetri. Kjo është realizuar duke përdorur threads.

Nje thread ose proces i lehte (LWP) është një njesi përdorimi e CPU që përbehet nga:

- program counter,
- një set regjistrash
- një hapesire stack.

Avantazhet e përdorimit të threads jane me të dukshme në sistemet multiprocesor, por edhe në sistemet me një CPU ka avantazhe të rëndësishme. Në fakt, mund të shfrytezohen kohet e vonesave/pritjeve gjate veprimeve I/O të një thread për të ekzekutuar një tjetër thread. Për të realizuar web server në rastin tone jane përdorur librarite pthread (POSIX thread) për të punuar me threads:

```
#include <pthread.h>
int pthread_create(pthread_t *thread, pthread_attr_t *attr,
void * (*start_routine)(void *), void *arg);
```

ne thread memorizohet thread ID, attr specifikon atributet e një thread-i të ri (vlera NULL bën të mundur të përdoren atributet default), start_routine është pointer tek funksioni që do ekzekutohet një thread-i i ri, arg është një pointer tek parametri që i kalohet funksionit start_routine; për të kaluar e shume parametra, mjafton të jepet një pointer për një strukture të dhënash.

Për menaxhimin e disa clients Jane përdorur threads, por nëpërmjet socket është e mundur të krijohet një pike lidhje midis një client që kërkon një burim dhe një serveri që e ofron ate burim. Permendem me pare se Jane objekte që krijohen për të realizuar një komunikim. Secili prej tyre ka karakteristika specifike, në lidhje me adresat dhe protokollet e përdorura. Perveç kësaj, socket jep mundesine e kryerjes se disa veprimeve (funksioneve), si për shembull lidhja me një kompjuter, pritja për lidhjen e dikujt, dërgimi dhe marrja e informacionit.

Në Linux, socket është një deskriptor që inicializohet me funksionin socket(), sipas sintakses int socket(int domain, int type, int protocol); jep një vlere të barabarte ose me të madhe se zero në rast suksesi, ose me të vogel se zero në rast gabimi. Jane të nevojshme ketu edhe headers #include <sys/types.h> e #include <sys/socket.h>.

Domain percakton një hapesire komunikimi, ose familjen e protokollit që do përdoret për komunikimin; type tregon llojin e socket që do krijohet; protocol tregon

protokollin e përdorur nga socket.

Funksionet me të rëndësishme për të punuar me sockets janë:

connect() për të vendosur një lidhje me një adresë lokale ose të largët;

bind() për të dhene një adresë dhe portë lokale tek socket, ku serveri do jete në degjim;

listen() për të vënë në degjim socket: përdoret pas bind();

send() për të transmetuar një mesazh të transmetuar një mesazh tek një tjetër socket

close() për të mbyllur një socket dhe për të cliruar burimin;

recv() për të marre një mesazh nga një socket

accept() për të pranuar lidhjet nga clients;

Në faqet e man mund të gjeni detaje të tjera.

Për të krijuar një lidhje të thjeshtë nga ana e client duhet:

1. krijuar një socket handle;
2. plotesuar struktura sockaddr me vlerat e nevojshme (te percaktuara në file-socket.h);
3. bërë lidhja nëpërmjet funksionit connect();
4. dërguar disa byte nëpërmjet send();
5. rilexuar të dhënat nëpërmjet recv();
6. mbyllur lidhja nëpërmjet close();

Nga ana tjetër, detyra e serverit është të jete në degjim në një portë dhe të administrojë lidhje të shumefishta nga ana e klienteve. Hapat e pare janë:

1. të krijohet një socket handle;
2. të plotesohet struktura sockaddr me vlerat e nevojshme;
3. të caktohet një adresë dhe një portë lokale me serverin nëpërmjet funksionit bind();
4. të fillojë degjin e lidhjeve të klienteve nëpërmjet funksionit listen();
5. të krijohen sockets për cdo lidhje klient nëpërmjet accept();

Dy shkaqet kryesore për të cilat një server nuk arrin të degjojë në një portë të caktuar, pra nuk realizohet një bind, janë se porta mund të jetë e zene ose mund të ketë një firewall që bllokon aksesin.

Nje web server i shkruar në C realizohet si me poshtë:

```
#include <stdio.h>      // Libraria Standard IO
#include <sys/socket.h> // Libraria Socket
#include <netinet/in.h> // Libraria e familjes se adresave internet
#include <arpa/inet.h> // Libraria e perkufizimeve për veprimet internet
#include <fcntl.h>       // Libraria për kontrollin e files
#include <stdlib.h>      // Libraria e per gjithshme
#include <string.h>       // Libraria për strings
#include <unistd.h>      // Libraria e simboleve standard për konstantet dhe llojet
#include <pthread.h>      // Libraria për menaxhimin e POSIX/thread
#include <dirent.h> // Libraria për files dhe arkivat

#define MAX_DIM 1024
#define SA struct sockaddr // struktura për adresat

// deklarohen funksionet kryesore që do të përdoren për
// web server

// funksion që bën të mundur dërgimin e përbajtjes se një file
void send_file(int,int);

void send_lista_directory(char *,int);
```

```
// funksion p\"er interpretimin e k\"erkesave
void interpreta_richiesta(int,char *);

void *lettura_richiesta(void *);

// funksion q\"e krijon nj\"e socket
void crea_socket(int);

// gjatesia e file
int lunghezza_file;

/***** MAIN *****/
int main(int argc,char **argv)
{
    // kontroll i portes q\"e do p\"erdoret nga socket
    int port;
    if (argc != 2)
    {
        printf ("Duhet dhene numri i portes \n");
        return 1;
    }

    // marr numrin e portes nga parametri i pare
    port = atoi(argv[1]);

    // e kaloj t\"e socket
    crea_socket(port);
    return 0;
}

// funksion q\"e krijon socket
void crea_socket(int port)
{
    int listenfd, connfd, clieLen = 0;

    // menaxhim i lidhjeve n\"e hyrje me POSIX/thread t\"e pavarur
    pthread_t thread;

    // deklaroj dy adresat client/server sipas struktures
    struct sockaddr_in servAddr,clieAddr;

    // vendos socket n\"e degjim
```

```
listenfd = socket(AF_INET,SOCK_STREAM,0);

// kontroll p\"er probleme n\"e hapjen e socket
if(listenfd < 0)
{
    printf("Gabim: socket nuk u hap. \n");
    exit(-1);
}

// kthen n\"e 0 adresen e memories q\"e tregohet nga &servAddr
memset(&servAddr,0,sizeof(servAddr));

//*** caktimi i adreses ****/

servAddr.sin_family=AF_INET;

// vendos n\"e degjim socket server n\"e t\"e gjitha adresat
servAddr.sin_addr.s_addr=htonl(INADDR_ANY);

// vendos porten e server q\"e e marr nga shell
servAddr.sin_port=htons(port);

// i shoqeroj porten serverit (per lidhjen) midis socket
// dhe portes dhe kontrolloj gabimet
if(bind(listenfd,(SA *)&servAddr,sizeof(servAddr))<0)
{
    printf("Gabim : Nuk lidhem dot me porten e caktuar \n");
    exit(-1);
}

// vendos n\"e degjim dhe kontrolloj gabimin
if(listen(listenfd,5)<0)
{
    printf("Gabim: Gabim n\"e degjim");
    exit(-1);
}

while(1)
{
    printf("\n Serveri n\"e degjim n\"e porten %d \n",port);
```

```
// socket pranon n\"e m\"enyr\"e t\"e papercaktuar lidhjet
connfd=accept(listenfd,(SA *)&clieAddr,&clieLen);

// dhe n\"e rast gabimi shfaq nj\"e mesazh gabimi
if(connfd<0)
{
    printf("Gabim: Nuk pranohet lidhja");
    continue;
}

// n\"e rast se nuk ka gabime, pranohet lidhja
// dhe tregohet adresa IP e konvertuar n\"e dhjetore XXX.XXX.XXX.XXX
printf("U vendos lidhja me %s\n",inet_ntoa(clieAddr.sin_addr));

// Krijon nj\"e POSIX/thread t\"e pavarur dhe ja shoqeron lidhjes
// ekzekuton n\"e thread leximin e k\"erkuar
pthread_create(&thread,NULL, lettura_richiesta,(void *)connfd);

// b\"en wait p\"er thread
pthread_join(thread,NULL);

// mbyll lidhjen e socket
close(connfd);
}

}

/** LEXIMI I KERKESES SE BROWSER **/


void *lettura_richiesta(void *arg)
{
    // identifikuesi i socket
    int sockid=(int)arg;

    // cakton memorien duke rezervuar nj\"e Buffer me 1024 Bytes
    char *buff=malloc(MAX_DIM);

    // i kthen n\"e 0 t\"e gjitha bytes t\"e buffer
    bzero(buff,MAX_DIM);

    // marrja nga ana e socket e t\"e dh\"enave q\"e d\"ergon Browser
```

```
// dhe i vendos n\"e buffer duke kontrolluar edhe p\"er gabime
if(recv(sockid,buff,MAX_DIM,0)<=0)
{
    printf("Gabim n\"e marrje.\n");
    exit(1);
}

// funksion q\"e interpreton k\"erkesat e browser
// q\"e i kemi n\"e Buffer
interpreta_richiesta(sockid, buff);
return NULL;
}

// interpretim i p\"ermbajtjes se Browser
void interpreta_richiesta(int sockid,char *buff)
{
    // identifikuesi p\"er file
    int fp;

    // deklaroj si char rreshtin q\"e i kaloj Browser, path, dhe Input
    char *res_line=malloc(MAX_DIM),path[100],*line;

    // vendos path = new q\"e do jete adresa si p.sh. l'htdocs di Apache
    strcpy(path,"net");

    // ndaj n\"e pjese buffer duke p\"erdorur read

    line=strtok(buff," ");

    // vendosem n\"e elementin e pare t\"e rreshtit
    line=strtok(NULL," ");

    // mund t\"e percaktoj path direkt nga browser
    if (strcmp(line,"/"))
    {

        // shtoj n\"e path p\"ermbajtjen e rreshtit
        strcat (path,line);

        // e printoj
        printf ("Path i faqeve web \"esht\"e : %s\n",path);
    }
}
```

```
// hap nj\"e file Read Only sipas path
if((fp=open(path,O_RDONLY,"r"))== -1)
{
    // n\"e rast se nuk gjjen kete path, d\"ergoj gabimin

    strcpy(res_line,"HTTP/1.1 404 Not Found\r\nContent_length:44\r\n\n
<html><body><h1>404 : Not Found</h1></body></html>\r\n\n");

    // dhe d\"ergoj p\"ermbajtjen e res_line tek browser n\"ep\"ermjet socket
    if(send(sockid,res_line,strlen(res_line),0)== -1)
    {
        printf("Error : Cannot send data.\n");
        return;
    }

    printf("File creato dinamicamente (errore) inviato \n");
    return;
}

}

else
// hap file nga arkivi i autorizuar. e ngjashme me
//htdocs n\"e apache
fp=open("net/index.htm",O_RDONLY,"r");

// marr gjatesine e file
lunghezza_file = lseek (fp,0,SEEK_END);

// e printoj
printf ("Dimensioni i faqes se transmetuar: %d %s\n",lunghezza_file," Byte");

// rikthehem n\"e fillim t\"e file
lseek (fp,0,SEEK_SET);

// n\"e t\"e dh\"enat q\"e d\"ergoj n\"e browser
// regjistroj tekstin q\"e p\"ermban kodin p\"er ngarkimin korrekt t\"e faqes
// dhe llojin e file me dimensionin
sprintf(res_line,"HTTP/1.1 200 OK\r\nContent_type:Application/Octet-stream,
text/html\r\nContent_length:%d\r\n\n",lunghezza_file);
```

```
// d\"ergoj tekstin e pare tek browser p\"er lidhjen e realizuar,
// llojin dhe dimensionin e file
if(send(sockid,res_line,strlen(res_line),0)==-1)
{
    printf(" Gabim n\"e d\"ergim");
    return;
}

// dhe n\"e vijim d\"ergoj edhe p\"ermbajtjen e file
// n\"ep\"ermjet nj\"e funksioni q\"e shfytet e socket
send_file(fp,sockid);

// d\"ergoj listen e arkivave dhe files
send_lista_directory(path,sockid);
return;
}

void send_lista_directory(char *path,int sockid)
{
// ndryshore q\"e mban listen e files
struct dirent **namelist;

// ndryshore ndihmese
char *StrAppoggio=malloc(MAX_DIM);

char res_line[MAX_DIM*2];
int n,i=0;

// marr p\"ermbajtjen e arkives dhe numrin
n = scandir(path, &namelist, 0, alphasort);

if (n < 0)
perror("scandir");
else {
sprintf(res_line,"%s","<br><h3>Lista e arkivave t\"e
cartelle della htdocs(Simile Apache):<h3><B><UL>");
while(i<n) {

// krijoj listen HTML me files q\"e i d\"ergohen browser
sprintf(StrAppoggio,"<a href=\"%s\"><li>",namelist[i]->d_name);
```

```
strcat(res_line, StrAppoggio);
strcat(res_line,namelist[i]->d_name);
strcat(res_line,"</a>");
i++;
}
strcat(res_line,"</UL></B>");

// d\"ergoj p\"ermbajtjen e file t\"e formatuar n\"e HTML
if(send(sockid,res_line,strlen(res_line),0)==-1)
{
printf(" Gabim d\"ergimi.");
return;
}
};

return;
}

// d\"ergim i file rresht p\"er rresht duke shfrytezuar socket
void send_file(int fp,int sockid)
{
int bytes;
char g_buff[MAX_DIM*2];
memset (g_buff,0,sizeof(g_buff));
while ((bytes=read(fp,g_buff,MAX_DIM*2))>0)
{
if (send(sockid,g_buff,bytes,0)==-1)
{
printf ("Gabim n\"e d\"ergimin e t\"e dh\"enave");
return;
}
}
return;
}
```

Fillimisht programi duhet kompiluar me komandën `gcc WebServ.c -o .../WebServ -lpthread` që e kemi në një skript Automake. Për ekzekutimin e programit mjafton të shkruajmë në shell `./WebServ` 8080, ku 8080 është numri i portës. Pas kësaj mund të shikojmë në funksion mini web server të krijuar: mjafton të ekzekutohet programi, të hapet browser dhe të lidhemi në adresen `http://192.168.1.4:8080`.



Figura 6.2. Web server

6.3.6 Chat

Chat është një aplikacion që u krijon mundesine përdoruesve të komunikojnë midis tyre në kohë reale. Eshte një aplikacion praktik i nevojshem dhe bën të mundur që përdoruesit që mund të ndodhen në lboratore në vende të ndryshme mund të bashkeveprojnë pa qene nevoja të përdorin telefonin ose ndonje mjet tjetër. Realizimi i këtij funksioni është bërë i mundur edhe për distribucionin tone, nëpërmjet një programi të thjeshtë në gjuhen C, që arrin të komunikojë me një tjetër kompjuter nëpërmjet socket, ashtu si edhe web server. Në fakt, janë shkruar dy programe në C, njëri për klientin dhe tjetri për server. Me poshte paraqitet kodi i programeve:

CLIENT

```
// header file për socket
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <netinet/in.h>

// header të tjerë
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
    int sd;                                // deskriptori i socket
    int bufsize = 1024, flag=0;   // dimensioni buffer stdin
    char *buffer = (char*) malloc(bufsize); // buffer për stdin
    struct sockaddr_in address;           // adresa dhe porta e serverit

    if (argc < 2) {
        printf("Perdorimi i Client : Client indirizzo_IP (i serverit për kontakt)\n");
        exit(0);
    }
```

```
}

if ((sd = socket(AF_INET, SOCK_STREAM, 0)) > 0)
    printf("Socket u krijua !\n"); // socket i krijuar pa gabime

// protokolli ARPA i Internet
address.sin_family = AF_INET;
// numri i portes se server (man htons) futet ne address
address.sin_port = htons(15000);
// IP server futet ne address

inet_pton(AF_INET, argv[1], &address.sin_addr);

/*
 * ne "ep\ermjet connect klienti lidhet me serverin dhe
 * realizohet komunikimi.
 * arg1: deskriptori i socket lokal
 * arg2: adresa e server ku lidhemi
 * arg3: dimensioni i struktures se adreses server
 *
 */

if (connect(sd, (struct sockaddr *)&address, sizeof(address)) == 0)
{
    printf("Lidhja u pranua me serverin: %s ...\n", inet_ntoa(address.sin_addr));
    printf("(quit per te e dale)\n");

}

/*
 * marrje dhe ne "ergim i mesazheve ne "ep\ermjet socket
 * arg1: deskriptori i socket
 * arg2: buffer ku vendoset dhe lexohet mesazhi
 * arg3: dimensioni i buffer
 * arg4: (flag)
 *
 */
do
{
    printf("Jep identifikuesin: \n");
    fgets(buffer, bufsize, stdin);
```

```
send(sd, buffer, bufsize, 0);
recv(sd, buffer, bufsize, 0);
if (!strcmp(buffer,"miresevjen Mikel"))
{
flag=1;
printf("%s\n",buffer);
}
}

while (flag!=1);

if (flag==1)
{
    while(1)
    {
        recv(sd, buffer, bufsize, 0);

        printf("Mesazhi u mor: %s\n", buffer);
        printf("Mesazhi p\"er d\"ergim: ");

        fgets(buffer, bufsize, stdin);
if( ! strcmp(buffer, "quit\n") )
{
        printf("\n\nDalje\n\n");
        return 0;
}

/*
 * transmeton mesazhin duke shkruar n\"e t\"e njejtin socket,
 * te njejta argumente t\"e recv
 *
 */
        send(sd, buffer, bufsize, 0);
    } //while
} //if
}

close(sd);      // mbylljet socket

return 0;
}
```

SERVER

```
// header file p\"er socket
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

// header t\"e tjera
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main()
{
    int sd, new_sd;
    socklen_t addrlen;
    int bufsize = 1024, flag=0;
    char *buffer = (char*) malloc(bufsize);

/*
 * strukture p\"er adresen IP te
 * server dhe porta n\"e t\"e cilen server \"esht\"e n\"e degjim
 * si dhe familjen e adresave t\"e p\"erdorura
 *
 */
    struct sockaddr_in address;

/*
 * strukture p\"er adresen IP
 * dhe porten e client (pas thirrje se accept,
 * netstat -at jep adresen e client dhe porten
 * q\"e do jete 32775 ose 32777 ....)
 * si dhe familjen e adresave t\"e p\"erdorura
 *
 */
}
```

```
struct sockaddr_in client_address;

system("clear");

/*
 * krijohohet socket ashtu si n\"e sock_client.c
 * arg1: AF_INET p\"erfaqeson protokollin ARPA
 *         (AF_INET = IPv4 internet protocol, IP e porta)
 * arg2: sock_stream jep nj\"e lidhje t\"e besueshme, n\"e sekuenc\"e,
 *       full duplex, nj\"e lidhje tcp
 * arg3: p\"erfaqeson protokollin, vlera zero i sherben sistemit p\"er t\"e zgjedhur
 *       ate me t\"e pershtatshem
 *
 */
if ((sd = socket(AF_INET,SOCK_STREAM,0)) > 0)
    printf("Socket u krijuar!\n");

/*
 * ashtu si p\"er client ( vedere ip(7) )
 *
*/
address.sin_family = AF_INET; // familja e adresave ARPA
address.sin_addr.s_addr = INADDR_ANY; // cdo adrese p\"er binding
address.sin_port = htons(15000); // numri i portes se degjimit server

/*
 * socket lokal i shoqerohet nj\"e proces n\"e pritje
 * t\"e lidhjeve nga cdo client n\"e porten 15000
 * arg1: deskriptori i socket sd krijuar me pare
 * arg2: struktura address q\"e ka porten e serverit
 *       dhe adresen IP t\"e server
 * arg3: dimensioni i struktures address
 *
*/

```

```

if ( bind(sd, (struct sockaddr *)&address, sizeof(address)) == 0 )
    printf("Procesi p\"er socket u krijua! \n");

/*
 * procesi server i krijuar \"esht\"e n\"e degjim
 * p\"er lidhje n\"e socket
 * arg1: deskriptori i socket
 * arg2: madhesia maksimale e radhes se lidhjeve
 *
 */

listen(sd, 3);

/*
 * accept (perdonur vetem me socket t\"e lidhur si SOCK_STREAM
 * q\"e jane p\"erdonur), merr lidhjen e pare t\"e radhes, krijon nj\"e socket t\"e ri
 * new_sd t\"e lidhur me t\"e njejtat veti t\"e sd dhe me t\"e dh\"enat
 * e client dhe lidh t\"e dy socket. new_sd kthehet nga accept
 * Serveri komunikon me client duke lexuar dhe shkruar ne
 * socket new_sd .
 * arg1: sd \"esht\"e deskriptori i gjeneruar nga thirrja socket
 *      n\"e client
 * arg2: adresa \"esht\"e ajo e serverit ku lidhet client
 *
 */

// addrlen q\"e sherben me pas p\"er accept
addrlen = sizeof(struct sockaddr_in);

// client_address merr t\"e dh\"enat e client
new_sd = accept(sd, (struct sockaddr *)&client_address, &addrlen);

if (new_sd > 0) // nuk ka gabime
{
    printf("%s u lidh ... \n", inet_ntoa(client_address.sin_addr));
    putchar(7);
}
else
    return 1;

do

```

```
{  
recv(new_sd, buffer, bufsize, 0);  
if( ! strcmp(buffer, "michele\n") )  
{  
printf("perdorues i njohur:%s\n",buffer);  
send(new_sd, "Mireseerdhe Michele", bufsize, 0);  
flag = 1;  
}  
else  
{  
if( ! strcmp(buffer, "quit\n") )  
flag = 2;  
else {  
printf("perdoruesi nuk u njoh\n");  
send(new_sd, "nuk je njohur. jep identifikativin ose quit p\"er t\"e dale", bufsize, 0);}}  
}  
while ((flag != 1) && (flag != 2));  
send(new_sd, "ciao", bufsize, 0);  
recv(new_sd, buffer, bufsize, 0);  
if (flag==1)  
{  
while(1)  
{  
printf("Mesazhi u mor: %s\n", buffer);  
printf("Mesazhi p\"er d\"ergim: ");  
fgets(buffer, bufsize, stdin);  
if( ! strcmp(buffer, "quit\n") ) //quit p\"er dalje  
{  
printf("\n\nDalje\n\n;a");  
return 0;  
}  
  
/*  
 * transmetim dhe marrje njesoj si n\"e sock_client  
 *  
 */  
send(new_sd, buffer, bufsize, 0);  
recv(new_sd, buffer, bufsize, 0);
```

```
    }
}

close(new_sd); // mbyllja e socket
close(sd);

return 0;
}
```

Udhezime për kompilimin dhe përdorimin e programit chat

Për kompilimin duhet të vendoseni në arkiven ku jane kodet dhe të ekzekutoni make. Do keni në output dy file të ekzekutueshem: "Server" dhe "Client". Për të lidhur dy kompjutera, njëri duhet të behet server, pra në të do ekzekutohet ./Server, kurse të tjetri ekzekutohet ./Client { adresa IP e server}. Pas kësaj krijohet lidhja dhe mund të përdoret programi chat.

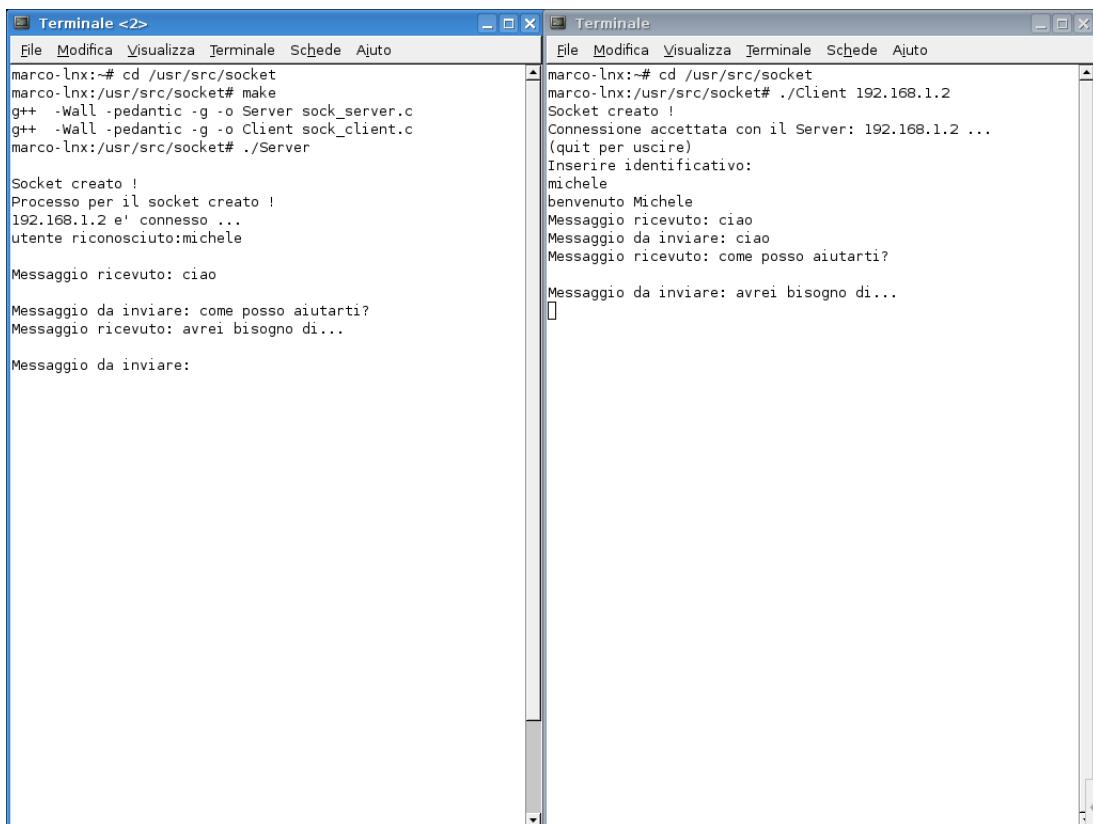
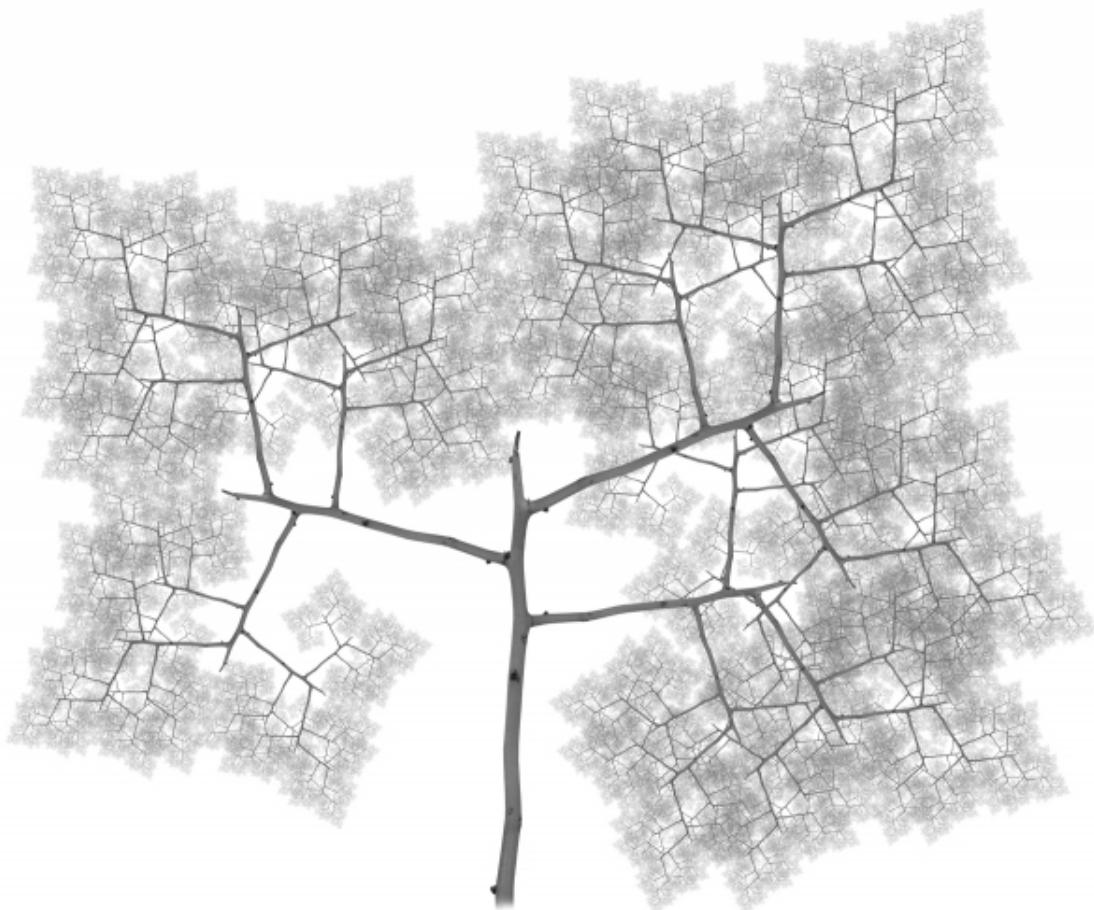


Figura 6.3. Chat

7 Projekte të realizuara



Në këtë kapitull paraqiten disa nga rezultatet e arritura nga grupe studentësh të koordinuar brenda grupit Hixos të Universitetit të Barit. Keto rezultate i gjeni të dokumentuara mirë në një wiki të grupit me adresë www.di.uniba.it/~hixos. Aty mund të gjeni edhe referenca dhe burime të tjera në rrjet, të cilat kanë bërë të mundur realizimin e projekteve.

7.1 Initrd

Ram disku fillestar është një filesystem i kompresuar që ngarkohet në RAM gjatë nisjes se kompjuterit menjëherë pas kernel. Nëpërmjet këtij filesystem behet i mundur konfigurimi i të gjitha moduleve për përdorimin e pajisjeve periferike.

Rasti konkret i studimit parashikonte kompilimin e Linux kernel dhe krijimin e një filesystem initramfs për montimin e një distribucioni kompakt në pendrive. Në veçanti, vëmendja u përqendrua në krijimin e initramfs, që ka si detyrë të vetme njohjen në mënyrë automatike të particionit në pendrive, ku është instaluar distribucioni, dhe e kthen atë në filesystem root nëpërmjet chroot.

7.1.1 Konfigurimi Buildroot

Shkarkoni versionin e fundit të buildroot nga faqja buildroot.uclibc.org¹. Kodi si gjithmonë duhet dekomprimuar në /usr/src.

```
cd /usr/src  
tar xvf buildroot-*.tar.bz2
```

Është e nevojshme të zhvendosemi në arkivin buildroot dhe të ekzekutojmë komandën make menuconfig për të filluar konfigurimin.

¹<http://buildroot.uclibc.org/>

```
cd buildroot  
make menuconfig
```

Në këtë fazë është e nevojshme të specifikohen paketa që do të përfshihen në initrd që po ndertojmë; perveç atyre që janë të përfshira default, per qellimet tona është e domosdoshme të kemi edhe busybox, udev dhe bash. Busybox e gjemë në menu Package Selection for the target, në të cilin do të kontrollojmë nëse janë të aktivuara edhe zërat Run BusyBox's own full installation dhe bash. Për të aktivuar udev zhvendosemi nga pozicioni aktual në nënmenu Hardware handling / blockdevices and filesystem maintenance dhe përzgjedhim udev, udev-utils dhe nga nënmenu Extra udev tools përzgjedhim zërin udev-libvolume_id. Ky opsjon është i nevojshem për të lejuar skriptin të gjejë në mënyrë korrekte të gjitha pajisjet e jashtme të lidhura në sistem nëpërmjet rregullave të udev.

Kthehemë në menunë kryesore dhe kalojmë në nënmenu Toolchain; që këtu aktivojmë opsjonin Enable large file (files > 2 GB) support ?, qe zgjidh problemin në fazen e komplimit.Pasi ruhen të gjitha zgjedhjet me komandën Save an Alternate Configuration File nga menuja kryesore, konfigurimi mund të quhet i përfunduar dhe kompilojmë duke përdorur komandën make.

7.1.2 Montimi i file imazh

Në përfundim të komplimit krijohet një file rootfs.i686.ext2 në arkivin binaries/u-clibc. Kopjojmë këtë file në arkivin e kodeve të Linux dhe zhvendosemi ne atë arkiv.

```
cp binaries/ulibc/rootfs.i686.ext2 /usr/src/linux-*  
cd /usr/src/linux-*
```

Ky file nuk është gjë tjeter vecse një file imazh, në të cilin kemi filesystem të initrd që krijuam. Për të modifikuar strukturen e initrd duhet kriuar arkivi initrams dhe

të montohet file rootfs.i686.ext2.

```
mkdir initramfs  
mount -o loop rootfs.i686.ext2 initramfs
```

7.1.3 Gjenerimi i nyjeve

Për të evituar problemet eliminojmë cdo file në arkivin initramfs/dev dhe krijojmë vetëm nyjet e nevojshme; popullimi i arkivit me nyjet e tjera do të behet nga udev në fazën e ekzekutimit.

```
rm -r initramfs/dev/*
```

Perdorim komandën mknod për të krijuar nyjet:

```
mknod initramfs/dev/null c 1 3  
mknod initramfs/dev/zero c 1 5  
mknod initramfs/dev/console c 5 1  
mknod initramfs/dev/tty1 c 4 1
```

7.1.4 Futja e skriptit dhe rregullat e udev

Në mënyrë që initrd të ekzekutojë korrektësish switch_root duhet të përfshije edhe një skript të emërtuar linuxrc, që sherben për njohjen dhe montimin e pajisjeve periferike të jashtme. Për funksionimin normal të skriptit nevojiten edhe dy rregulla të udev që ndodhen në file 60-persistent-input.rules dhe 60-persistent-storage.rules.

Vijon përbajtja e file linuxrc:

```
#!/bin/sh  
mount -o remount,rw /  
mount -t proc proc /proc  
mount -a  
for i in /etc/init.d/S??* ;do  
    # Ignore dangling symlinks (if any).  
    [ ! -f "$i" ] && continue
```

```

    case "$i" in
        *.sh)
            # Source shell skript for speed.
            (
                trap - INT QUIT TSTP
                set start
                . $i
            )
            ;;
        *)
            # No sh extension, so fork subprocess.
            $i start
            ;;
    esac
done
# Riconosce univocamente la penna usb ed esegue il mount
declare -a TOKENS
declare -a DEVS
SCSI_DEV_DIR="/sys/class/scsi_device"
TOKENS=(`ls -l /dev/disk/by-uuid | cut -d " " -f29`)
DEVS=(`ls $SCSI_DEV_DIR`)
NUM_TOKENS=${#TOKENS[@]}
NUM_DEVS=${#DEVS[@]}
if [ $NUM_TOKENS == 0 ]; then
    printf "Errore: l'implementazione di ls -l potrebbe essere cambiata\n"
    exit 5
fi
if [ $NUM_DEVS == 0 ]; then
    printf "Errore: l'implementazione di sysfs potrebbe essere cambiata\n";
    exit 5
fi
i=0
while [ $i -lt $NUM_TOKENS ]; do
    if [ $($(( $i % 2)) -ne 0) ]; then
        DEV=$(printf "%{TOKENS[$i]}")
        sed -e 's|..|..||' -e 's/[0-9]//'
        #controlla se $DEV e' una periferica rimovibile
        j=0
        while [ $j -lt $NUM_DEVS ]; do
            REMOVABLE=$(cat "$SCSI_DEV_DIR/${DEVS[$j]}/device/block"
-----$DEV/removable" 2> /dev/null)
            # ok la periferica e' rimovibile
            if [ "$REMOVABLE" == 1 ]; then
                cd /dev/disk/by-uuid
                mount ${TOKENS[$i]} /mnt
                printf "periferica ${DEV} montata..\n"
                umount /proc
                # pivot_root per passare alla nuova root
                /sbin/pivot_root /mnt /mnt/initrd
                # passa il controllo alla shell
                cd /
                /bin/bash
            fi
            let j=j+1
        done
    fi
    i=i+1
done

```

```
let i=i+1
done
```

Listing 7.1. linuxrc

Supozojmë se 3 file ndodhen brenda arkivit /usr/src:

```
cp /usr/src/linuxrc initramfs/linuxrc
cp /usr/src/ 60-persistent-input.rules initramfs/60-persistent-input.rules
cp /usr/src/ 60-persistent-storage.rules initramfs/60-persistent-storage.rules
```

Direktiva rootdelay=10 dërgohet tek kernel nëpërmjet GRUB dhe sjell një pauzë fillestare prej 10 sekondash; kjo pritje është e nevojshme per t'i dhënë kohe udev për popullimin e arkivit /dev/disk/by-uuid/ me symlink për pajisjet periferike të lidhura në sistem dhe per të populluar arkivin /sys/class/devices me informacionet mbi pajisjet e ndryshme.

```
declare -a TOKENS
declare -a DEVS
SCSI_DEV_DIR="/sys/class/scsi_device"
TOKENS=(`ls -l /dev/disk/by-uuid | cut -d " " -f28 `)
DEVS=(`ls \$SCSI_DEV_DIR`)
[ ... ]
if [ \$NUM_TOKENS == 0 ]; then
    printf "Errore: l'implementazione di ls -l potrebbe essere cambiata\n"
    exit 5
fi
if [ \$NUM_DEVS == 0 ]; then
    printf "Errore: l'implementazione di sysfs potrebbe essere cambiata\n";
    exit 5
fi
```

Listing 7.2. linuxrc

Më sipër shikoni një pjese të kodit të linuxrc ku, nëpërmjet komandës ls, lexohen nga arkivat e cituara emrat e pajisjeve të njohura. Aty behet edhe një kontroll mbi output të ls për të verifikuar nëse stringat janë lexuar sic duhet; në rast gabimi sugjerohet rikontrollimi i parametrave të komandës, sepse mund të ndodhe për shembull që versioni i komandës ls që përdoret nga sistemi prites dhe ai i Buildroot të jene të

ndryshme dhe po keshtu edhe output i ketyre komandave, Në vijim skripti ben dy cikle leximi: cikli i jashtem lexon gjithe pajisjet kurse cikli i brendshem /emphdevice të sistemit. Sapo gjendet një pajisje që i perket një device të jashtem, kjo pajisje montohet dhe ekzekutohet *switch-root*, duke u zhvendosur mbi te.

7.1.5 Modalitetet e ngarkimit të initrd

Pasi arritem në këtë pike, kemi dy mundësi: të përfshijme initrd në kernel ose të ngarkojmë initrd nëpërmjet menu.lst. Opcioni i pershkruar me poshtë është opzioni i dyte që parashikon ndarjen e initrd nga kernel. Duke zgjedhur këtë mundësi, initrd vendoset në arkivin /boot në pendrive dhe ngarkohet në memorie menjëherë pas kernel. Avantazhi kryesor i kësaj mënyrë është se mund të modifikojmë initrd pa qene nevoja të rikompilojmë kernel.

Pasi ngarkohet në memorie initrd, kontrolli i kalon /linuxrx; nëse ky file nuk ndodhet në initramfs ateherë na jep një gabim dhe ekzekutimi bllokohet.

Pasi i kryejme pa gabime hapat e mesipërme, duhet të smontojmë file rootfs.i686.ext2, ta kompresojmë, ta riemërtojmë dhe ta kopjojmë ne arkivin boot të distribucionit tone.

```
umount rootfs.i686.ext2
gzip rootfs.i686.ext2
mv rootfs.i686.ext2.gz initrd.img
cp initrd.img /mnt/mvux/boot
```

Pasi komprimohet initrd, mjafton të shtojmë në file menu.lst një rresht që tregon adresen e tij, si me poshtë:

```
boot ...
initrd /boot/initrd.img
```

7.1.6 Problemet e hasura

Transferimi i arkivit root nga RAM tek pendrive mund të behet në dy menyra:

- pivot_root

Është një komande që merr në input një arkiv, që do behet root, dhe arkivin në të cilin do transferohet e vjetra.

```
/sbin/pivot\_root /mnt /mnt/initrd
```

Kjo komande nuk jep gabime.

- switch_root

Është një komande që merr në input arkivin që do mbaje root dhe komandën e parë që do ekzekutohet pasi të behet transferimi.

```
exec /sbin/switch\_root /mnt /bin/bash
```

Komanda switch_root nederpritet duke dhënë gabimin not rootfs, pra nuk behet transferimi i arkivit root tek pendrive. Kjo ndodh sepse ka disa kufizime përfunksionimin korrekt të programit; këto shprehen në pjesen e kodit C të marre nga kodi i switch_root:

```
if (lstat("/init", &st1) || !S_ISREG(st1.st_mode) || statfs("/", &stfs) ||
    (stfs.f_type != RAMFS_MAGIC && stfs.f_type != TMPFS_MAGIC) ||
    getpid() != 1)
{
    bb_error_msg_and_die("not_rootfs");
}
```

Listing 7.3. switch_root.c

Testi vertetohet nëse vlejne kushtet e mëposhtme:

- file /init duhet të jete një file i rregullt (regular file); nuk mund të jete i llojit link ose device;
- root filesystem / duhet të jete i llojit ramfs ose tmpfs;
- procesi që therret switch_root duhet të ketë pid 1.

Kushti i parë nuk plotesohet sepse file /init nuk ekziston. Ky problem zgjidhet lethesisht duke riemërtuar /.bashrc në /init; në këtë mënyrë skripti që kemi perqatitur ekzekutohet direkt. Kushti i dyte vertetohet sepse initrd është i llojit ramfs. Kushti i trete është ai që pengon ekzekutimin e skriptit. Pas shume provash nuk u bë e mundur që procesi /init të ketë pid 1. Për këtë u vendos përdorimi i pivot_root dhe jo i switch_root.

7.2 Nfs

Nfs është një akronim për emërtimin Network File System. Nfs zakonisht përdoret për montimin e një filesystem që ndodhet në një kompjuter të largët.

7.2.1 Konfigurim i Kernel

Versioni i përdorur për kernel është 2.6.27.4. Jane përfshire në kernel perberesit e mëposhtëm:

- Networking support --> Networking options --> tcp/ip networking -->
ip: kernel level autoconfiguration

qe ben të mundur konfigurimin e rrjetit në fazën e nisjes se sistemit. Në këtë opsjon janë të aktivuara edhe nenopsionet:

- DHCP support
- BOOTD support
- RARP support

ka tre menyra të ndryshme për të realizuar një lidhje (pra një sistem diskless). Në vijim do të shohim se është zgjedhur një shperndarje statike e adresave IP tek klientet.

- Networking support --> Networking options -->
Network packet filtering framework (netfilter)

per të evituar (ne fazën e shperndarjes dinamike të adresave IP) problemin e mëposhtëm:

socket: Address family not supported by protocol - make sure CONFIG_PACKET (Packet

in your kernel configuration!

Failed to bring up eth0

- File systems --> Kernel automounter support
- File systems --> Network file Systems --> NFS client support --> NFS client support for NFS version 3
- File systems --> Network file Systems --> NFS client support --> NFS client support for NFS version 4
- File systems --> Network file Systems --> NFS server support --> NFS server support for NFS version 3
- File systems --> Network file Systems --> NFS server support --> NFS server support for NFS version 4

Suporti për server NFS është i detyrueshem vetëm për kernel të kompjuterave server.

7.2.2 Networking

- Nderfaqesi i rrjetit i njohur nga kernel dhe plotesisht funksional
- Net-tools ²

Në rastin e një ndarje të thjeshte të arkivave (pra jo në rastin diskless) sigurohuni që keni ekzekutuar:

`ifconfig lo up`

per të filluar sherbimin që gjen hostname lokale. Nje parakusht për funksionimin e server nfs eshte:

²<http://www.linuxfromscratch.org/blfs/view/svn/basicnet/net-tools.html>

"Name service (host name lookup) should be working before any NFS services are started."

7.2.3 Klienti diskless

Grub

Në manualin e grub mund të lexoni si me poshtë:

13.2.6 ifconfig

Command: ifconfig [--server=server] [--gateway=gateway] [--mask=mask]

[--address=address]

Configure the IP address, the netmask, the gateway, and the server address of a network device manually. The values must be in dotted decimal format, like '192.168.11.178'. The order of the options is not important.

This command shows current network configuration, if no option is specified

Si general command mund të futet në çfaredo pike të *menu.lst* dhe në rreshtin e komandës. Kjo zgjidhje rezultoi e papershtatshme sepse nuk lejonte montimin e root në largesi nëpërmjet NFS.

Si parameter i command-line kernel mund të konfigurohej karta e rrjetit e klientit nfs:

ip=ip_del_client:dns:gateway:subnet:hostname_del_client:network_interface

Parametra të tjere të nevojshem janë:

```
root=/dev/nfs
nfsroot=ip_del_server:/data/nfs
rootdelay=10
```

Ku:

- root specifikon se root është montuar me filesystem NFS
- nfsroot specifikon vendndodhjen e arkivit që do montohet si root
- roottdelay specifikon numrin e sekondave të pritjes para se të kërkohet root

Perfundimisht në *menu.lst* opsioni për boot diskless do jete i mëposhtmi (ne rastin tone):

```
root      (hd0,0)
kernel   /boot/vmlinuz  root=/dev/nfs rw  nfsroot=10.1.1.1:/disklessroot ip=10.1.1.10:10.1.1.1:10.1.1.1:255.255.255.255
boot
```

Listing 7.4. menu.lst

Problemi i hasur

Me konfigurimin e meparshem, sistemi jepte një kernel panic:

```
VFS: Cannot open root device "nfs" or unknown-block (0,255)
Please append a correct "root=" boot option
Kernel panic - not syncing: VFS: Unable to mount root fs on unknown block (0,255)
```

Zgjidhja gjendet në konfigurimin e kernel. Duke aktivuar si modul të brendshem

```
File systems --> Network file Systems --> NFS client support -->
Root filesystem on NFS root
```

i jepet mundësia sistemit të montojë filesystem root me një arkiv të ndare me filesystem NFS. Opsiuni i cituar me sipër do jete i dukshem vetëm nëse është përfshire opsiuni ”ip: kernel level autoconfiguration” (sic specifikohet në seksionin ”Paramkushtet”).

Kuriozitete

Në parametrin ip të command-line mund të specifikohet përdorimi i protokollit dhcp:

```
ip=dhcp
```

Kjo mundësi u perjashtua përfaktin se një server mund të ospitojë shume sisteme dhe në baze të IP të ndaje një root ose një tjetër. Për këtë arsyen nisja e sistemit diskless do jete me IP statike.

7.2.4 Serveri NFS

Në arkivin që ndahet me klientin diskless MVUX do të kemi gjithe filesystem MVUX, perveç arkivit boot, që duhet të jetë në pendrive. Pendrive me distribucionin MVUX, sic e thame, do të mbaje ekskluzivisht arkivin boot me grub dhe kernel.

Zgjidhja e pare

Si zgjidhje kryesore u zgjodh konfigurimi dhe përdorimi i një serveri NFS direkt nga distribucioni aktual (Ubuntu). Për detajet teknike mund t'i referoheni dokumentacionit zyrtar të komunitetit ku diskutohet mbi Konfigurimin e NFS Server.

Zgjidhja e dyte

Si zgjidhje alternative u mendua të përdorej një paketë nga debian repository që instalohet në MVUX, në mënyrë që të evitojen probleme te lidhura ngushte me konfigurimin e paketës:

```
apt-get install -d --reinstall nfs-kernel-server
```

Ndermjet varesive të tjera të kësaj pakete gjejmë edhe paketen "libblkid1". Kjo e fundit varet nga "libuuid1" e cila varet nga "passwd", që në vijim varet nga paketat

e nevojshme për login (paketa pem). Kjo e ben konfigurimin shume me kompleks se nevojat tona. Për këtë arsye zgjidhja e dyte nuk u konsiderua.

Zgjidhja e trete

Se fundmi, një zgjidhje tjetër është instalimi i serverit NFS në distribucionin MVUX duke përdorur paketën platform-independent. Për një funksionim normal të serverit NFS duhet përdorur vetëm portmap me paketën nfs-utils.

nfs-utils

Paketa mund të shkarkohet në internet. Vilon komplili i nfs-utils (duke instaluar gjithe package-dev të nevojshme për ekzekutimin e konfigurimit):

```
./configure --prefix=/usr/src/nfs_compilato/  
--sysconfdir=/etc --disable-nfsv4 --disable-gss
```

- me ane të prefix mund të specifikohet një output i ndryshem nga ai default dhe për të kopjuar files të gjeneruar në distribucionin MVUX
- disable-nfsv4 sherben për të përdorur v3 që nuk kërkon autentikimin e përdoruesit, që në rastin tone nuk është i nevojshem
- disable-gss për të disaktivuar suportin RPCSEC GSS (RPC Security) dhe kerberos v5, që nuk janë të nevojshme në rastin tone

```
sudo make  
sudo make install
```

Në vijim, kopjohen nga arkivi i output file të gjeneruara në pendrive me distribucionin MVUX. Siç citohet në README të paketës:

```
"To use the 'gss' support you must have kerberos-5 development  
libraries installed. Otherwise use '--disable-gss'"
```

portmap

Në UNIX demoni portmap (porte TCP/111 dhe UDP/111) mban:

- listen e programeve RPC të regjistruar
- porten dhe protokollin në të cilin janë në degjim

NFS përdor thirrjet me procedure të largët (RPC) për funksionimin; komanda portmap është i nevojshem për të shoqëruar kërkesat RPC me sherbimet e duhura. Proseset RPC njoftojnë portmap kur fillojnë funksionimin, duke dhënë numrin e portes që po monitorojnë dhe numrat e programeve RPC që janë gati të sherbejne. Sistemi klient pastaj kontakton portmap në server me një numer të veçante programi RPC. Në këtë pike, portmap riadreson klientin tek numri i portes korrekte ne mënyrë që të komunikojë për sherbimin e deshiruar. Paketa e përdorur mund të shkarkohet në internet. Siç specifikohet në README të paketës:

```
There's no "./configure", just use "make".
```

Kopojmë komandat portmap, pmap_set dhe pmap_dump në pozicionet e duhura në distribucionin MVUX (/sbin).

File të exports

Krijojmë një file në /etc:

```
nano /etc/exports
```

Ky file përmban gjithe arkivat e ndara në NFS me të drejtat respektive. Cdo rresht është një deklarim arkivi që ndahet në rrjet dhe respekton sintaksen e mëposhtme:

```
directory machine1(option11,option12) machine2(option21, option22)
```

ku:

- directory është arkivi që ndahet
- machine është adresa IP e kompjuterit që merr të drejtat dhe option janë opzionet e mundshme:
 - ro: readonly
 - ose rw: read & write
- no_subtree_check: përdoret në rastin kur ndajme vetëm një nenpeme të arkivit
- sync: default përdoret një sjellje asinkrone dhe kjo mund të sjelle korruptimin e file. Ky opzion modifikon në largesi njekohesisht me modifikimet në kompjuterin lokal. Duhet bërë kujdes që të mos ketë shume diferenca (jo me shume se pak minuta) midis ores se klientit dhe ores se serverit
- no_root_squash: nenkupton se cdo përdorues që futet në arkivin e ndare në NFS ka të njejtat privilegje që do kishte root në server

7.2.5 Skripti për autoboot

Parakushtet për këtë faze janë komandat "Make" dhe "Install" të distribucionit MVUX. Duke analizuar readme të paketës nfs-utils shohim shenimin e mëposhtëm:

"This nfs-utils packages does not provide any skripts for starting various daemons as most distributions replace them with their own, so any skripts we package would not get much testing."

Kjo na ben të mendojmë të marrim skriptin që ndodhet në distribucion (/etc/init.d/nfs-kernel-server), por kjo nuk funksionon. Në fakt, skripti nuk ekzekuton asgje nese përdoret në distribucionin MVUX. Duke kërkuar në rrjet gjeni disa skripte të gatshme që korrespondojnë me skriptet e nisjes se LFS (linux from scratch)

* Dekomprimojmë paketën në distribucionin MVUX

```
cd blfs-bootscripts-20080816/
make install-nfs-server
```

Krijojmë file për konfigurimin /etc/sysconfig/nfs-server:

```
sudo mkdir /etc/sysconfig
sudo nano /etc/sysconfig/nfs-server
```

dhe bejme konfigurimet që vijojnë:

```
NFSV4=no
PORT="2049"
PROCESSES="8"
QUOTAS="no"
KILLDELAY="10"
```

I njejti problem është edhe me skriptet që administrojnë portmap:

```
make install-portmap
```

7.2.6 NFS dhe ndarja e pajisjeve

Duke rikujtuar se në Linux cdo gjë është një file dhe NFS është një filesystem përrjetin, lind pyetja:

A mund të eksportohet me NFS një pajisje (per shembull një kamera rrjeti) në mënyrë që ajo të përdoret edhe nga larg njesoj sikur të ishte në kompjuterin lokal?

Pergjigjja është jo sepse sherbimi NFS ndan vetëm arkivat dhe filesystems. Arkivat e pajisjeve janë arkiva të veçante qe përdoret nga sistemi dhe që njihet fale drivers

specifike që ndodhen në kernel të serverit. Pra edhe nëse ndahen files, pajisja nuk do të funksiononte nga një kompjuter klienti. Edhe nëse ndaheshin files të pajisjes, do lindte një problem ”device busy”. Ndryshon puna në rastin e ndarjes se pajisjeve të memorizimit sepse mjafton te montohen në server dhe të ndahet nëpërmjet NFS arkivi i duhur.

7.2.7 Nderfaqesi wifi

Parakushtet

Sipas dokumentacionit të kernel:

In order to use this driver, you will need a firmware image for it.

Keshtu që u mor **firmware** specifik për nderfaqesin *wireless*.

You will also very likely need the Wireless Tools in order to configure your card.

Si instrumenta ndihmes u instaluan në /sbin (duke respektuar dipendencat):

- ifrename
- iwconfig
- iwevent
- iwgetid
- iwlist
- iwpriv

- iwspy

Nje tjetër citim interesant i dokumentacionit të kernel është ky që vijon:

It is recommended that you compile this driver as a module (M) rather than built-in (Y). This driver requires firmware at device initialization time, and when built-in this typically happens before the filesystem is accessible (hence firmware will b\"e unavailable and initialization will fail). If you do choose to build this driver into your kernel image, you can avoid this problem by including the firmware and a firmware loader in an initramfs.

dhe në baze të ketyre sugjerimeve u gjeten zgjidhjet që vijojnë.

Kompilimi i driver si modul

Në rastin specifik u përdor një karte rrjeti wireless Intel pro/set wireless 2200 bg. Driver për këtë pajisje ka nevojë për një firmware të posaçëm. Ky kusht eshte një kufizim i forte për nisjen e një sistemi diskless, sepse për të ngarkuar një firmware duhet akses në filesystem gjatë një faze (boot i kernel) kur kjo nuk është e mundur. Zgjidhja e parë konsiston në komplimin e driver si modul për nderfaqesin e rrjetit dhe përfshirjen në /etc/fstab të arkivit te larget, duke përdorur:

```
<server>:</path/of/dir> </local/mnt/point> nfs <options> 0 0
```

ku:

- <server-host> korrespondon me hostname ose adresen IP të server që eksporton (specifikuar në /etc/exports) filesystem
- </path/of/directory> path i arkivit të eksportuar

- </local/mount/point> specifikon ku montohet (ne filesystem lokal) arkivi i eksportuar. Arkivi duhet të ekzistojë para se të lexohet (gjate boot) file /etc/fstab perndryshe nuk montohet dot.
- nfs specifikon llojin e filesystem që montohet
- <option> korrespondon me të gjitha opsonet klasike të montimit për një filesystem (p.sh. rw,rw...)

Kjo zgjidhje është e papershtatshme për të përdorur burimet e nevojshme për boot me NFS (pra me sisteme diskless)

Perfshirja e firmware në kernel si blob

Pozicionohuni në zërin e mëposhtëm të kernel:

`Device drivers --> Generic driver options`

Në këtë seksion është e mundur të konfigurohen një sere konfigurimesh interesante:

`Userspace firmware loading support`

Aktivoni zërin e specifikuar

`Include in-kernel firmware blobs in kernel binary`

Aktivoni zërin e specifikuar. Në këtë mënyrë është e mundur të përfshini një firmware në kernel si **blob** dhe të evitoni nevojen për te aksesuar filesystem ose pëta ngarkuar në initramfs (initrd).

`External firmware blobs to build into the kernel binary`

Në këtë zë specifikohet emri (bashke me zgjatimin) e firmware. Ky duhet të jetë kopjuar me parë (versioni binar) në arkivin ”firmware” që ndodhet në kernel e dekomprimuar.

Firmware blobs root directory

Në këtë zë specifikohet arkivi (duke filluar si root nga arkivi i dekomprimuar i kernel) ku do të merret firmware i specifikuar në zërin me sipër.

Kjo zgjidhje na eviton një nga të dy problematikat: mundësine për të pasur një nderfaqe wireless funksionale gjatë fazes se boot të kernel. Problemi i cili ben të pamundur një sistem diskless me nderfaqe wireless është se nuk ekziston mundësia të lidhet nderfaqesi me një ssid para montimit të filesystem.

7.2.8 Perdorimi i initramfs

Nje parakusht për funksionimin korrekt të kësaj zgjidhje është përfshirja në konfigurimin e kernel e zërit të mëposhtëm:

General Setup --> Initial RAM filesystem and RAM disk (initramfs/initrd) support

Ka disa driver (si në rastin tone) që në faze inicializimi ngarkojnë një firmware. Fatmirësisht, në fazën e boot, initramfs na jep mundësine të aksesojmë firmware. Hapat për të realizuar këtë zgjidhje janë:

- Konfiguro driver të kartes wireless si e brendshme në kernel (dhe jo si modul)
- Specifiko se çfare duhet të përfshije kernel në initramfs
- duke u nisur nga root i kernel krijohet file usr/default-files:

```
dir /dev 0755 0 0
nod /dev/console 0600 0 0 c 5 1
dir /root 0700 0 0
```

- Krijo strukturen për suportin e initramfs

```
sudo mkdir usr/root usr/root/firmware usr/root/sbin usr/root/bin
usr/root/sys
```

dhe në vijim kopjo firmware në usr/root/firmware

- Krijo një skript për ngarkimin e firmware

Krijo file file usr/root/sbin/hotplug:

```
#!/bin/sh

load_firmware() {
    if [ ! -e /firmware/$FIRMWARE ]; then
        echo $FIRMWARE not found >/dev/console
        return
    fi
    echo Loading $FIRMWARE into $DEVPATH >/dev/console
    mount -t sysfs sysfs /sys
    echo 1 > /sys/$DEVPATH/loading
    cat /firmware/$FIRMWARE > /sys/$DEVPATH/data
    echo 0 > /sys/$DEVPATH/loading
    umount /sys
}

case $ACTION in
    add)
        case "$FIRMWARE" in

```

```
        """ )  ;;
*) load_firmware  ;;

esac
;;
esac
```

Listing 7.5. usr/root/sbin/hotplug

- përdor një shell posix (Portable Operating System Interface for Unix) për ekzekutimin e skriptit

```
sudo apt-get install -d --reinstall busybox-static
```

Duke u nisur nga arkivi i pendrive ekzekutoni:

```
sudo cp /path/dove/si/trova/busybox usr/root/bin
cd usr/root/bin
sudo ln -s busybox sh
```

- Modifiko disa konfigurime që specifikojnë source dhe root në initramfs

```
sudo make menuconfig
```

Pozicionohu ne

```
General setup --> Initial RAM filesystem and RAM disk
(initramfs/initrd) support
```

dhe jep vlore për zërin "Initramfs source file(s)" me "usr/default-files usr/root"
Konfigurimet e CONFIG_INITRAMFS_ROOT_UID dhe CONFIG_INITRAMFS_ROOT_GID
ndryshojnë në baze të përdoruesit që kryen boot të sistemit.

- Kompilo kernel

```
sudo make  
sudo make bzImage  
sudo make modules
```

7.3 Busybox

BusyBox: thika zvicerane e sistemeve Linux embedded

BusyBox është një software i lire që kombinon versione të ndryshme të komandave Linux standard në një file të vogel, i cili është i ekzekutueshem. Në fakt, BusyBox ofron pjesen me të madhe të programeve ndihmese, që zakonisht përdoruesit i gjejne në sistemet GNU/Linux. Programet e BusyBox kane opsione të reduktuara në krahasim me versionet origjinale të tyre. Sidoqofte, opsonet e ofruara janë të mjaftueshme për përdoruesin mesatar. BusyBox ofron një ambient të kompletuar për cdo sistem të permasave të vogla ose për sisteme embedded. Ai është shkruar duke marre parasysh problemet e optimizimit të hapesires dhe të burimeve të kufizuara. Nje tjetër karakteristike e BusyBox është modulariteti, me të cilin është e mundur te përfshihen/perjashtohen lehtesisht komanda ose karakteristika në fazën e komplimit. Kjo e ben me të thjeshte personalizimin e sistemit embedded. Perdorimi është i thjeshte: mjafton të shtohen disa device në arkivin /dev, disa arkiva konfigurimi në /etc dhe një kernel Linux.

Fillimisht i shkruar nga Bruce Perens në vitin 1996, qellimi i BusyBox ishte të ndertohej një sistem i plote vetëm në një floppy. Ky mund të përdorej edhe për të rregulluar sisteme ekzistuese, por edhe për të instaluar Debian GNU/Linux. Më pas u bë një standard de facto për pajisjet Linux embedded dhe në instalimin e distribucioneve. Më parë cdo file i ekzekutueshem kërkonte disa KB, me BusyBox,

ku kombinohen me shume se dyqind programe se bashku, nevojitet shume me pak hapesire memorie.

Disa nga komandat e suportuara janë: adduser, chroot, cp, fdisk, ifconfig.

Programet e përfshira në BusyBox mund të ekzekutohen thjesht duke shtuar emrin e tyre si parameter gjatë ekzekutimit të BusyBox:

```
/bin/busybox cp p1 p2
```

Zakonisht, arkivi i ekzekutueshem linkohet (me link simbolik ose me hard link) me komandën e deshiruar.

Tani, BusyBox administrohet nga Denys Vlasencome licence GNU GENERAL PUBLIC version 2.

7.3.1 Applet

Ideja e "thikes zvicerane" si mjet që nuk zë shume vend është karakteristika me evidente e BusyBox: vetëm një file binar me shume linke simbolike, të emërtuar me emrat e komandave që zëvendësohen nga BusyBox. Pra, BusyBox përcakton se cilin applet të ekzekutojë duke ekzaminuar vektorin argv[0].

Applet e BusyBox janë me të vogla në krahasim me ato origjinaler sepse permasat e kodit janë qellimi kryesor i projektimit. Shume applet të BusyBox janë zhvilluar duke u nisur nga SingleUnix Specification version 3, të tjerat janë derivate të NetBSD dhe të ngjashme. Applet shpesh rishkruhen dhe rirealizohen për të zgjedhur dimensionin e tyre në favor të kursimit të memories.

Kompilimi i BusyBox është mjaft i thjeshte. Procesi i kompilimit përfshin nxjerrjen e burimit, konfigurimin me komandën "make menuconfig" dhe ekzekutimin nëpërmjet komandës "make".

Disa opsione të tjera të komandës make janë: "make defconfig" për të aktivuar të gjitha karakteristikat standard, "make allnoconfig" për të dizaktivuar të gjitha

opsionet, ”make oldconfig” për të përdorur arkivin .config me të fundit.

Nga menuconfig, cdo applet mund të selektohet pavaresisht nga të tjerat. Shume applet kanë një nënmenu karakteristikash që mund të dizaktivohet për të kursyer hapesire dhe burime.

7.3.2 Si të shtojmë një applet në BusyBox

Duke u nisur nga udhezuesi i vogel në faqen e BusyBox, u bë e mundur të shtohet një program personal në formen e applet në BusyBox.

Programi që u shtua quhet Chmusic.

Chmusic është një program që ben të mundur ndryshimin e të dhenave *meta* në *inode* të një ose me shume arkivave duke bërë te mundur edhe rirenditjen automatike të hierarkise se arkivave.

riordinamento automatico della gerarchia di cartelle.

7.3.3 Shtimi i Chmusic në BusyBox

Udhezimet e per gjithshme për shtimin e një applet të ri, që ndodhen në faqen e BusyBox, janë si me poshtë:

To add a new applet to busybox, first pick a name for the applet and a corresponding CONFIG_NAME. Then do this:

- Figure out wherë in the busybox source tree your applet best fits, and put your source code therë. Be sure to use APPLET_main() instead of main(), wherë APPLET is the name of your applet.
- Add your applet to the relevant Config.in file (which file you add it to determines wherë it shows up in ”make menuconfig”). This uses the same general format as the linux kernel’s configuration system.

- Add your applet to the relevant Makefile.in file (in the same directory as the Config.in you chose), using the existing entries as a template and the same CONFIG symbol as you used for Config.in. (Don’t forget ”needlrbm” or ”needcrypt” if your applet needs libm or libcrypt.)
- Add your applet to ”include/applets.h”, using one of the existing entries as a template. (Note: this is in alphabetical order. Applets are found via binary search, and if you add an applet out of order it won’t work.)
- Add your applet’s runtime help text to ”include/usage.h”. You need at least appname_trivial_usage (the minimal help text, always included in the busybox binary when this applet is enabled) and appname_full_usage (extra help text included in the busybox binary with CONFIG_FEATURE_VERBOSE_USAGE is enabled), or it won’t compile. The other two help entry types (appname_example_usage and appname_notes_usage) are optional. They don’t take up space in the binary, but instead show up in the generated documentation (BusyBox.html, BusyBox.txt, and the man page BusyBox.1).
- Run menuconfig, switch your applet on, compile, test, and fix the bugs. Be sure to try both ”allyesconfig” and ”allnoconfig” (and ”allbareconfig” if relevant).

Më poshtë raportohen hapat për shtimin e applet Chmusic në BusyBox.

Vini re se hapat e sugjeruara në faqen e BusyBox jepin disa gabime gjatë kompilimit.

Te gjitha problemet u zgjidhen nëpërmjet hapave të mëposhtëm:

1. Modifikim i arkivit arch/\$ARCH si me poshtë:

```
CFLAGS += \$(call cc-option,-march=i386 -mpreferred-stack-boundary=2,)
```

nese kemi një arkitekturë i386

```
CFLAGS += $(call cc-option,-march=arm -mcpu s3c2442  
-mpreferred-stack-boundary=2,)
```

nese kemi një arkitekturë arm.

2. Krijohet arkivi NOME_APPLET brenda BusyBox.
3. Kirjohet arkivi Kbuild brenda arkivit të applet si me poshtë:

```
lib-y:=  
lib-$(CONFIG_CHMUSIC) += ./hixos-common/hixos.o  
lib-$(CONFIG_CHMUSIC) += chmusic.o
```

qe specifikon arkivin që duhet përdorur gjatë kompilimit.

4. Krijo arkivin Config.in në arkivin e krijuar me sipër si vijon:

```
menu "Hixos _CHMUSIC"  
config CHMUSIC  
bool "changes_file_music_attributes"  
default y  
help  
Enable options and features which are not essential.  
endmenu
```

Listing 7.6. Config.in

Vini re se stringa pas CONFIG duhet të jetë njesoj me emrin e arkivit të krijuar në hapin e meparshem.

5. Modifiko file Config.in të BusyBox duke shtuar instrukzionin me poshtë:

```
"source chmusic/Config.in"
```

ne këtë pjese të kodit:

```
source archival/Config.in
source chmusic/Config.in
source coreutils/Config.in
source console-tools/Config.in
source debianutils/Config.in
source editors/Config.in
source findutils/Config.in
source init/Config.in
source loginutils/Config.in
source e2fsprogs/Config.in
source modutils/Config.in
source util-linux/Config.in
source miscutils/Config.in
source networking/Config.in
source printutils/Config.in
source mailutils/Config.in
source procps/Config.in
source runit/Config.in
source selinux/Config.in
source shell/Config.in
source sysklogd/Config.in
```

6. Modifiko Makefile në arkivin e BusyBox duke shtuar këtë rresht kodi:

```
"chmusic/ "
```

ne këtë pjese:

```
core-y := \
applets/ \
libs-y := \
archival/ \
archival/libunarchive/ \
chmusic/ \
console-tools/ \
coreutils/ \
coreutils/libcoreutils/ \
debianutils/ \
e2fsprogs/ \
editors/ \
findutils/ \
init/ \
libbb/ \
libpwdgrp/ \
loginutils/ \
mailutils/ \
miscutils/ \
modutils/ \
networking/ \
networking/libiproute/ \
networking/udhcp/ \
printutils/ \
procps/ \
runit/ \
```

```
selinux/ \
shell/ \
sysklogd/ \
util-linux/ \
util-linux/volume\_id/ \
```

7. Modifiko include/applets.h në arkivin e BusyBox duke shtuar:

```
USE\_CHMUSIC(APPLET\_NOEXEC(chmusic, chmusic, \_BB\_DIR\_BIN, \_BB\_SUID\_NEVER, chmusic))
```

ne pjesen e kodit që ka lidhje me përdorimin e applet të BusyBox, duke respektuar rendin alfabetik. Në të kundërt, applet nuk do funksionoje.

8. Modifiko include/usage.h në arkivin e BusyBox duke shtuar:

```
\#define chmusic\_trivial\_usage \
"chmusic-[OPTIONS]...-FILE"

\#define chmusic\_full\_usage "\n\n"
" Il comando chmusic permette di cambiare i tag in un file musicale.\n \
"\nOptions:" \
"\n-a-cambia il nome dell'autore" \
"\n-d-cambia il nome del disco" \
"\n-t-cambia del titolo" \
"\n-y-cambia l'anno di pubblicazione" \
```

Listing 7.7. include/usage.h

9. Riemërto main të kodit të applet si EMRI_APPLET_main në këtë mënyrë:

```
int chmusic\_main(int argc, char **argv)
```

ne vend të

```
main(int argc, char **argv)
```

10. Shto, para perkufizimit të main, instruksionin:

```
int chmusic\_main(int argc, char **argv) MAIN\_EXTERNALLY\_VISIBLE;
```

11. Shto në chmusic.h në arkivin e applet instruksionin:

```
#include "libbb.h"
```

ne pjesen e direktivave të përfshirjes.

12. Modifiko në hixos.h në arkivin e applet kodin e mëposhtëm sipas arkitekturës se përdorur:

```
/** Numero syscall chtag p\”er_i386*/
#define _SYS\_ctag_325
/** Numero syscall retag_p\”er_i386*/
#define _SYS\_retag_326

/** Numero syscall chtag p\”er_arm*/
#define _SYS\_ctag_361
/** Numero syscall retag_p\”er_arm*/
#define _SYS\_retag_362
```

Listing 7.8. hixos.h

qe ka lidhje me përcaktimin e numrave për syscall perkatese.

Vini re se numrat e syscall ndryshojnë sipas arkitektures.

Vini re se hapi 12) u krye për të shtuar applet chmusic, por nuk është i nevojshem për applet të tjera.

Perfundime

Më tej, sugerojmë ekzekutimin e komandës make allnoconfig për të pastruar konfigurimin fillestar dhe për të evituar probleme papajtueshmerie midis opsjoneve. Pas ketyre hapave u ekzekutua komand make menuconfig për të parë nëse applet chmusic ishte realisht e pranishme në zerat e BusyBox.

Për të testuar funksionimn e chmusic nga BusyBox u shtuan, me të njejten mënyrë si me pare, këto applet të nevojshme për ekzekutimin e chmusic:

- Addmusic
- Orderby
- Statmusic

Pasi u shtuan këto applet të reja, u verifikua funksionimi korrekt i chmusic nga BusyBox.

7.3.4 Problemet e hasura dhe zgjidhjet

Me shtimin e applet të reja, që përdorin hixos.h dhe hixos.c, zgjodhëm të spostonim këto në hixos-common. Si rrjedhim:

- u modifikuan include të chmusic, statmusic, orderby duke ndryshuar path për hixos.h
- u modifikua kbuild duke ndryshuar path për hixos.o.

Lindi nevoja të ripërcaktohen, për shkak të gabimeve compile-time, perkufizimet e syscall në hixos.h. U zëvendësua instruksioni i vjetër:

```
int syscall(int, ...);
```

me

```
extern long int syscall (long int \_\_sysno, ...) \_\_THROW;
```

Më pas, në disa nga applet e ngarkuara për funksionimin e chmusic, funksionet komande_gabim dhe help ishin të përcaktuara.

Për të zgjidhur këtë problem, funksionet u riemërtuan në orderby.c dhe statmus.c

Më tej, në menu konfigurimi të BusyBox duhet aktivuar zëri:

```
Build Option -> Build Shared Libbusybox
```

sepse në rast të kundërt nuk gjenden libraritë e nevojshme për ekzekutimin.

Në përfundim, duhet bërë kujdes të mos kompilohet BusyBox në mënyrë statike. Për të bërë këtë nuk duhet aktivuar zëri:

```
Build Option -> Build Static
```

7.4 Instaluesi

7.4.1 Objektivi

Objektivi i këtij rasti studimi është realizimi i një instaluesi grafik, i cili ben të mundur instalimin e distribucionit MVux në një pendrive me qellimin për ta bërë me të lehte dhe intuitiv procesin e instalimit.

Ideja

Për realizimin kemi marre si referim një instalues grafik të përdorur nga Linux Slackware (i shkruajtur në bash). Kemi analizuar kodet nëpërmjet të cileve është e mundur të fshihet, modifikohet dhe të shtohet kod për ta pershtatur me qellimin tone.

Kushtet

- një kompjuter me një distribucion Linux (p.sh. Ubuntu).
- të pakten 1Gb hapesire të lire në hard diskun e kompjuterit.
- Nje pendrive me të pakten 256 Mb
- Lidhje me internetin për të shkarkuar librarinë ”dialog”.

`apt-get install dialog`

Questa libreria permette di visualizzare l’interfaccia dell’installer. Kjo librari mundeson të shihet nderfaqesi i instaluesit.

7.4.2 Installer

Në arkivin SETUP_MVUX kemi krijuar arkivin SourcePackage.

Ky arkiv përbën të gjitha paketat, të komplikuara, që mund të instalohen. U zgjodh të bëhej me parë kompilimi dhe jo gjatë instalimit për të mos e ngarkuar shume këtë proces.

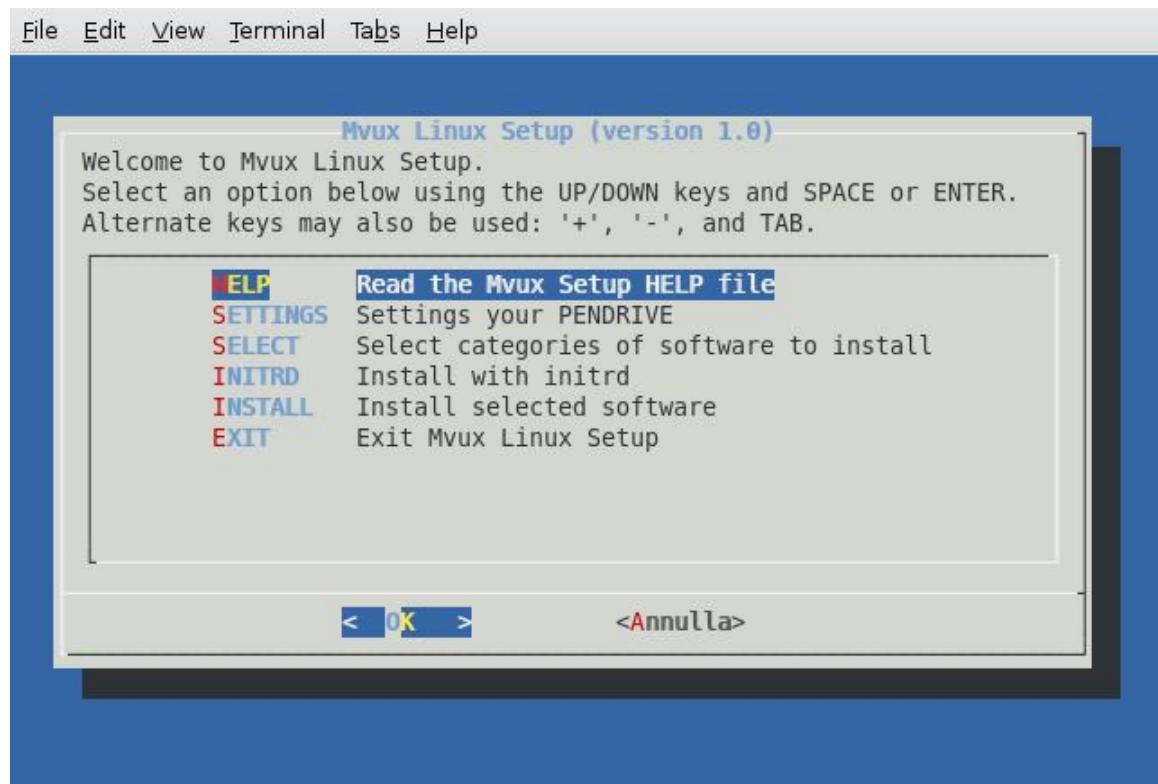
- Kernel - ver. 2.6.23
- Bash - ver. 3.2
- Grub - ver. 0.97

- CoreUtils - ver. 6.9.90
- Utilinux - ver. 2.12p
- Sysvinit - ver. 2.86
- e2fprog - ver. 1.40.8
- Emacs
- Busybox - ver. 1.13.1
- Initrd
- Skripti scompatta.sh: për shpjegimin shikoni me sipër
- Skripti scompattadeb: për shpjegimin shikoni me sipër

7.4.3 Setup i skriptit

Ky është skripti baze që starton instaluesin duke shfaqur menunë kryesore nga thirren nënmenu të tjera që jepin mundësine për veprime të ndryshme.

Menu Kryesore



Setup i skriptit

```
#!/bin/sh
TMP=/var/log/setup/tmp
if [ ! -d \$TMP ]; then
    mkdir -p \$TMP
fi
PATH="$PATH:."
export PATH;
ROOT_DEVICE=$(mount | grep "on /media" | cut -f 1 -d ' ')
if [[ \$ROOT_DEVICE = "" ]]; then
    dialog --title "MVUX_LINUX_SETUP" \
    --msgbox "\n\n\nInsert your PENDRIVE and restart Mvux Linux Setup" 10 60
    exit
else
    echo $ROOT_DEVICE > rootdev
    umount $ROOT_DEVICE
```

```

fi
while [ 0 ]; do
    dialog --title "Mvux_Linux_Setup_(version_1.0)" \
--menu \
"Welcome_to_Mvux_Linux_Setup.\n\
Select_an_option_below_using_the_UP/DOWN_keys_and_SPACE_or_ENTER.\n\
Alternate_keys_may_also_be_used: '+', '-', and TAB."_18_72_9 \
"HELP"_"Read the Mvux Setup HELP file" \
"SETTINGS"_"Settings your PENDRIVE" \
"SELECT"_"Select categories of software to install" \
"INITRD"_"Install with initrd" \
"INSTALL"_"Install selected software" \
"EXIT"_"Exit Mvux Linux Setup" 2>$TMP/MainMenu

if [ ! $?==0 ]; then
rm -f $TMP/MainMenu#$TMP/SeT*
exit
fi
MAINSELECT=`cat $TMP/MainMenu`
rm $TMP/MainMenu

## Start checking what to do. Some modules may reset MAINSELECT to run the
## next item in line.

if [ "$MAINSELECT" == "HELP" ]; then
SeTfdHELP
fi

if [ "$MAINSELECT" == "SETTINGS" ]; then
SeTFfileSystem
fi

if [ "$MAINSELECT" == "SELECT" ]; then
SeTPKG
fi

if [ "$MAINSELECT" == "INITRD" ]; then
MvuxInitrd
fi

if [ "$MAINSELECT" == "INSTALL" ]; then

```

```
...MvuxInstall
...
if [ "$MAINSELECT" == "EXIT" ]; then
if mount | grep /mnt/hd1>/dev/null >/dev/null; then
umount /mnt/hd
fi
if [ -r $TMP/SetSERIES ]; then
rm $TMP/*
fi
rm rootdev
break
fi

done #end of main loop
#sync
```

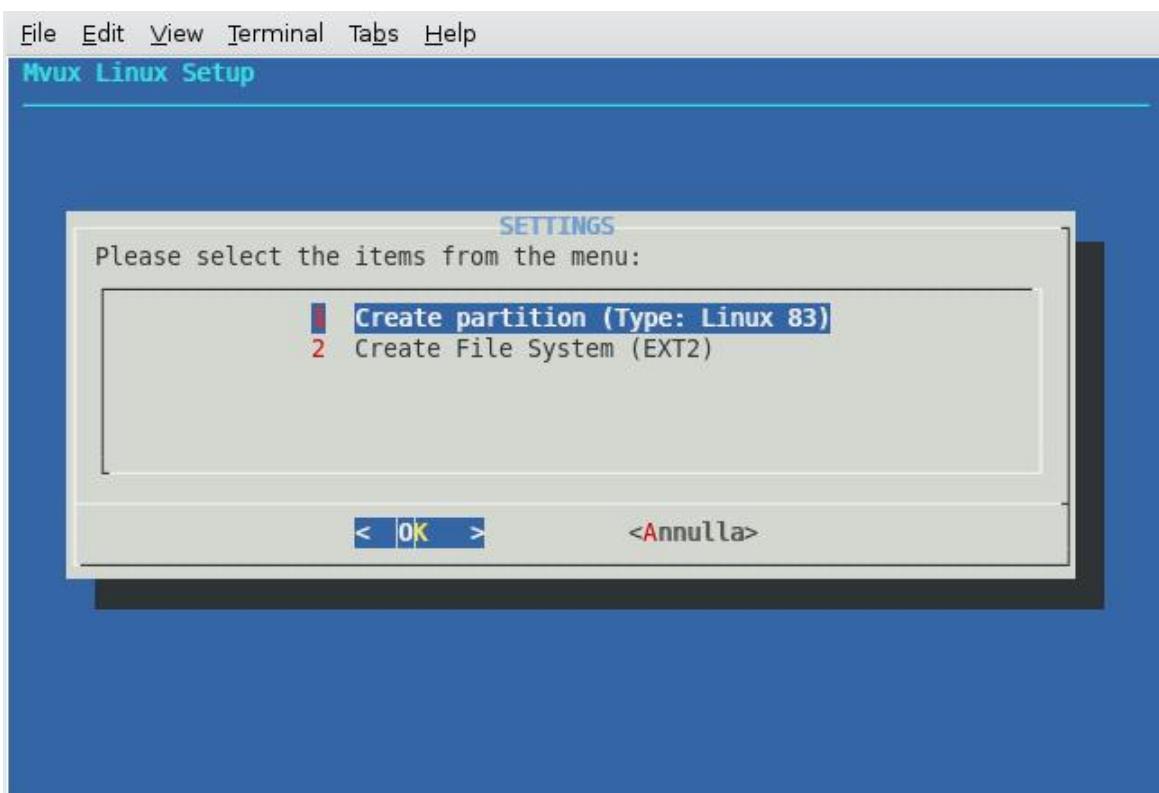
Listing 7.9. setup

Skripti MVUX_SETTINGS

Ky skript përfaqeson nënmenunë e lidhur me zërin ”SETTINGS” në menunë kryesore. Në këtë nënmenu përban dy veprime, ajo e particionimit dhe ajo e krijimit të filesystem. Për të dyja veprimet janë krijuar skriptet e duhura.

- Script MVUX_CFDISK: permette il partizionamento della pen drive. lejon ndarjen e pen drive.
- Script MVUX_FileSystem_ext2: permette la creazione del file system ext2.lejon krijimin e file system

Nenmenu SETTINGS



Skripti MVUX_SETTINGS

```
#!/bin/sh
#
TMP=/var/log/setup/tmp
if [ ! -d $TMP ]; then
    mkdir -p $TMP
fi
dialog --backtitle "Mvux_Linux_Setup" \
--title "SETTINGS" --menu \
"Please select the items from the menu:" \
12 70 5 \
"1" "Create_partition_(Type:_Linux_83)" \
"2" "Create_File_System_(EXT2)" \
2>$TMP/setting

if [ ! $? = 0 ]; then
    rm $TMP/setting
    exit
fi
```

```
SOURCE_MEDIA="`cat $TMP/setting`"  
rm -f $TMP/setting  
  
if [ "$SOURCE_MEDIA" = "1" ]; then  
    MVUX_CFDISK  
elif [ "$SOURCE_MEDIA" = "2" ]; then  
    MVUX_FILESYSTEM_ext2  
fi
```

Listing 7.10. MVUX_SETTINGS

Skripti MVUX_CFDISK

```
#!/bin/sh  
TARGET=`cat rootdev`  
if mount | grep $TARGET 1> /dev/null 2> /dev/null ; then  
    umount $TARGET  
    sleep 2  
    TARGET=`cat rootdev | cut -c 1-8`  
    cfdisk $TARGET  
    TARGET=`cat rootdev`  
    umount $TARGET  
    SetFileSystem  
  
else  
    TARGET=`cat rootdev | cut -c 1-8`  
    cfdisk $TARGET  
    TARGET=`cat rootdev`  
    umount $TARGET  
    SetFileSystem  
fi
```

Listing 7.11. MVUX_CFDISK

Skripti MVUX_FileSystem_ext2

```
#!/bin/sh  
TARGET=`cat rootdev`  
if mount | grep $TARGET 1> /dev/null 2> /dev/null ; then  
    umount $TARGET  
    mkfs.ext2 $TARGET  
    sleep 3
```

```

SeTFileSystem

else
    mkfs.ext2 $TARGET
    sleep 3
    SeTFileSystem
fi

```

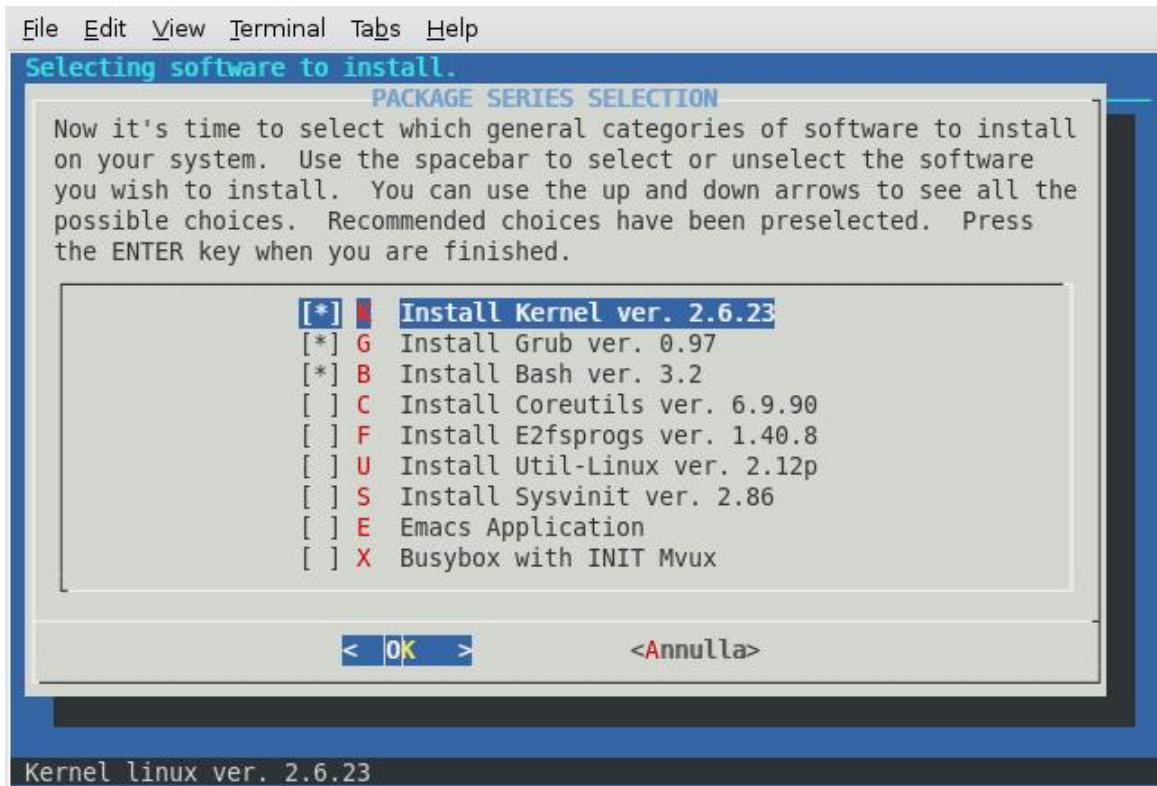
Listing 7.12. MVUX_CFDISK

Skripti SETPKG

Ky skript përfaqeson nënmenunë e lidhur me zërin SELECT të menuse kryesore. Në këtë nënmenu është e mundur të seleksionohen paketat që instalohen. Në veçanti seleksionimi është organizuar keshtu:

- Linux Base System: është i seleksionuar si default. Ai përfshin paketat baze të distribucionit MVux: Kernel, Grub, Bash.
- Paketa opsionale: përfaqesojnë komanda të dobishme për të përdorur distribucionin e Mvux: CoreUtils, UtilLinux, Sysvinit, e2fprog
- nuk është e selektuar si default pasi mund të mos jete e nevojshme. Për të instaluar Emacs duhen selektuar të gjitha paketat e tjera opsionale ose me busybox.
- Busybox with INIT Mvux: një distribucion i përdorur për sistemet embedded që përdor INIT të Mvux

NENMENU SELECT PACKAGE



Skripti SETPKG

```
#!/bin/sh
TMP=/var/log/setup/tmp

if [ ! -d $TMP ]; then
    mkdir -p $TMP
fi

rm -rf $TMP/tmpskript $TMP/series

# ----- Configurazione iniziale
STATO_K="on"
STATO_G="on"
STATO_B="on"
STATO_C="off"
STATO_F="off"
STATO_U="off"
STATO_S="off"
```

```

STATO_E=" off"
STATO_X=" off"

if [ -e $TMP/K ]; then
K='cat $TMP/K'
if [ -z $K ]; then
STATO_K=" off"
else
STATO_K=" on"
fi
fi

if [ -e $TMP/G ]; then
G='cat $TMP/G'
if [ -z $G ]; then
STATO_G=" off"
else
STATO_G=" on"
fi
fi

if [ -e $TMP/B ]; then
B='cat $TMP/B'
if [ -z $B ]; then
STATO_B=" off"
else
STATO_B=" on"
fi
fi

if [ -e $TMP/C ]; then
C='cat $TMP/C'
if [ -z $C ]; then
STATO_C=" off"
else
STATO_C=" on"
fi
fi

```

```
if [ -e $TMP/F ]; then
F="`cat $TMP/F`"
if [ -z $F ]; then
STATO_F="off"
else
STATO_F="on"
fi
fi
```

```
if [ -e $TMP/U ]; then
U="`cat $TMP/U`"
if [ -z $U ]; then
STATO_U="off"
else
STATO_U="on"
fi
fi
```

```
if [ -e $TMP/S ]; then
S="`cat $TMP/S`"
if [ -z $S ]; then
STATO_S="off"
else
STATO_S="on"
fi
fi
```

```
if [ -e $TMP/E ]; then
E="`cat $TMP/E`"
if [ -z $E ]; then
STATO_E="off"
else
STATO_E="on"
fi
fi
```

```
if [ -e $TMP/X ]; then
X="`cat $TMP/X`"
```

```

if [ -z $X ]; then
    STATO_X="off"
else
    STATO_X="on"
fi
fi

cat << EOF > $TMP/tmpskript
dialog --backtitle "Selecting software to install." \\
--title "PACKAGE_SERIES_SELECTION" --item-help --checklist \\
"Now it's time to select which general categories of software \\
to install on your system...\\"
Use the spacebar to select or unselect the software you wish to \\
install...\\
You can use the up and down arrows to see all the possible choices...\\
Recommended choices have been preselected...\\
Press the ENTER key when you are finished." \\
20 75 9 \\
"K" "Install_Kernel_ver_2.6.23" \$STATO_K "Kernel_linux_ver_2.6.23" \\
EOF

cat << EOF >> $TMP/tmpskript
"G" "Install_Grub_ver_0.97" \$STATO_G "Grub_ver_0.97" \\
EOF

cat << EOF >> $TMP/tmpskript
"B" "Install_Bash_ver_3.2" \$STATO_B "Bash_shell_ver_3.2" \\
EOF

cat << EOF >> $TMP/tmpskript
"C" "Install_Coreutils_ver_6.9.90" \$STATO_C "Coreutils_ver_6.9.90" \\
EOF

cat << EOF >> $TMP/tmpskript
"F" "Install_E2fsprogs_ver_1.40.8" \$STATO_F "E2fsprogs_ver_1.40.8" \\
EOF

cat << EOF >> $TMP/tmpskript
"U" "Install_Util-Linux_ver_2.12p" \$STATO_U "Util-Linux_ver_2.12p" \\
EOF

```

```

cat << EOF >> $TMP/tmpskript
"S" "Install_Sysvinit_ver_2.86" \$STATO_S "Sysvinit_ver_2.86" \\
EOF

cat << EOF >> $TMP/tmpskript
"E" "Emacs_Application" \$STATO_E "The_E_series_is_a_Emacs_application" \\
EOF

cat << EOF >> $TMP/tmpskript
"X" "Busybox_with_INIT_Mvux" \$STATO_X "The_X_series_is_a_Busybox_with_INIT_Mvux" \\
EOF

cat << EOF >> $TMP/tmpskript
2> $TMP/series
EOF

. $TMP/tmpskript

if [ ! $? = 0 ]; then
    rm -rf $TMP/series $TMP/tmpskript
    exit
fi

INSTSETS="`cat $TMP/series | tr -d '\n'`"
INSTSETS="`echo $INSTSETS | tr '\042' '#'`"
INSTSETS="`echo $INSTSETS | tr ',' '#'`"
rm -rf $TMP/series $TMP/tmpskript
echo "$INSTSETS" > $TMP/SeTSERIES

# Toglie tutte le grattelle      Heq t\ "e gjitha %%%%%%%%%%%%%%
cat $TMP/SeTSERIES | tr "#" "\>" > $TMP/pulisci           pastron

# Output del file in modo verticale      output
cat $TMP/pulisci | tr "\>" "\n" > $TMP/verticale

# Cattura le lettere in modo da installare i pacchetti selezionati dall'utente
cat $TMP/verticale | grep "K" > $TMP/K # CATTURA LA LETTERA "K" Kernel
cat $TMP/verticale | grep "G" > $TMP/G # CATTURA LA LETTERA "G" Grub
cat $TMP/verticale | grep "B" > $TMP/B # CATTURA LA LETTERA "B" Bash
cat $TMP/verticale | grep "C" > $TMP/C # CATTURA LA LETTERA "C" Core Utils

```

```

cat $TMP/verticale | grep "F" > $TMP/F # CATTURA LA LETTERA "F" e2fsprogs
cat $TMP/verticale | grep "U" > $TMP/U # CATTURA LA LETTERA "U" Util-linux
cat $TMP/verticale | grep "S" > $TMP/S # CATTURA LA LETTERA "S" Sysvinit
cat $TMP/verticale | grep "E" > $TMP/E # CATTURA LA LETTERA "E" EMACS
cat $TMP/verticale | grep "X" > $TMP/X # CATTURA LA LETTERA "X" BUSYBOX

```

Listing 7.13. SETPKG

Ndihma e skriptit

Ky skript ka një HELP në menunë kryesore që jep një seri ndihmash për të vazhduar me instalimin.

Skripti verificaldd.sh

Shpjegimin e këtij skripti e gjeni me sipër.

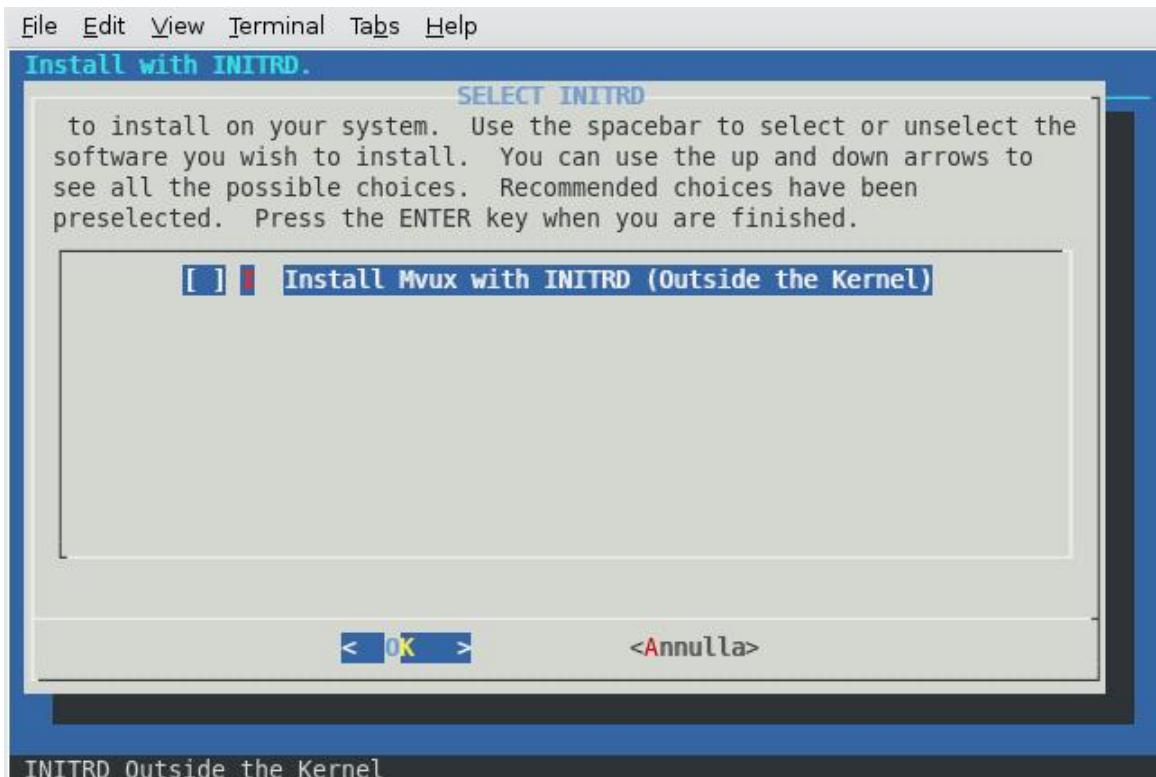
Skripti ldd.sh

Shpjegimin e këtij skripti e gjeni me sipër.

Skripti MvuxInitrd

Ky skript shoqërohet me zërin INITRD nga menuja kryesore dhe ben të mundur përdorimin e initrd në pendrive.

NENMENU SELECT INITRD



Skripti MvuxInitrd

```
#!/bin/sh
TMP=/var/log/setup/tmp

if [ ! -d $TMP ]; then
    mkdir -p $TMP
fi
cd $TMP
touch INITRD
# ----- Configurazione iniziale
STATO_INITRD="off"

if [ -e $TMP/INITRD ]; then
    INITRD=`cat $TMP/INITRD`
    if [ -z $INITRD ]; then
        STATO_INITRD="off"
    else
        STATO_INITRD="on"
```

```

fi
fi

cat << EOF > $TMP/tmpinitrd
dialog --backtitle "Install-with_INITRD." \\
--title "SELECT_INITRD" --item-help --checklist \\
"\\"
to_install_on_your_system.\\
Use_the_spacebar_to_select_or_unselect_the_software_you_wish_to\\
install.\\
You_can_use_the_up_and_down_arrows_to_see_all_the_possible_choices.\\
Recommended_choices_have_been_preselected.\\
Press_the_ENTER_key_when_you_are_finished." \\
20 75 9 \\
" I" " Install_Mvux_with_INITRD_(Outside_the_Kernel)" \\
\$STATO_INITRD "INITRD_Outside_the_Kernel" \\
EOF

cat << EOF >> $TMP/tmpinitrd
2> $TMP/series
EOF

. $TMP/tmpinitrd

if [ ! $? = 0 ]; then
rm -rf $TMP/series $TMP/tmpinitrd
exit
fi

INSTSETS="`cat $TMP/series | tr -d "\n"`
INSTSETS="`echo $INSTSETS | tr "\042" "#`"
INSTSETS="`echo $INSTSETS | tr , "#`"
rm -rf $TMP/series $TMP/tmpinitrd
echo "$INSTSETS" > $TMP/SeTINITRD

# Toglie tutte le graticelle Heg t\ "e gjitha kartelat
cat $TMP/SeTINITRD | tr "# " > $TMP/pulisciINITRD

# Output del file in modo verticale

```

```
cat $TMP/pulisciINITRD | tr " " "\n" > $TMP/verticaleINITRD

# Cattura le lettere in modo da installare i pacchetti selezionati dall'utente
cat $TMP/verticaleINITRD | grep "I" > $TMP/INITRD
# CATTURA LA LETTERA "I" MVux whit INITRD
```

Listing 7.14. MvuxInitrd

Script Mvux_Install:

Ky skript shoqërohet me zërin INSTALL nga menuja kryesore dhe lejon të nisesh instalimin e paketave të zgjedhura në pendrive. Skripti Mvux_Install

```
#!/bin/sh
TMP=/var/log/setup/tmp

if [ -r $TMP/SeTSERIES ]; then

    if [ -r $TMP/SeTINITRD ]; then
        INITRD="`cat $TMP/INITRD`"
    fi

    K="`cat $TMP/K`"
    G="`cat $TMP/G`"
    B="`cat $TMP/B`"
    C="`cat $TMP/C`"
    F="`cat $TMP/F`"
    U="`cat $TMP/U`"
    S="`cat $TMP/S`"
    E="`cat $TMP/E`"
    X="`cat $TMP/X`"

    # controlla se è stata selezionata una configurazione valida
    # K G B
    # K G B C
    # K G B C F
    # K G B C F U
    # K G B C F U S
    # K G B C F U S E
    # X
```

```

# X E
# controlla se e' stata selezionata una configurazione valida con l' INITRD
# K G B C F U S + INITRD
# K G B C F U S E + INITRD
# X + INITRD
# X E + INITRD

if ( [ -n "$K" ] && [ -n "$G" ] && [ -n "$B" ] && [ -z "$C" ] && [ -z "$F" ]
&& [ -z "$U" ] && [ -z "$S" ] && [ -z "$E" ] && [ -z "$X" ] && [ -z "$INITRD" ] ) ||
( ( [ -n "$K" ] && [ -n "$G" ] && [ -n "$B" ] && [ -z "$X" ] &&
[ -z "$E" ] ) && [ -z "$INITRD" ] && ( [ -n "$C" ] || [ -n "$F" ] || [ -n "$U" ]
|| [ -n "$S" ] ) ) ||
( [ -n "$K" ] && [ -n "$G" ] && [ -n "$B" ] && [ -n "$C" ] && [ -n "$F" ]
&& [ -n "$U" ] && [ -n "$S" ] && [ -n "$E" ] && [ -z "$X" ] && [ -z "$INITRD" ] ) ||
( [ -n "$X" ] && [ -z "$K" ] && [ -z "$G" ] && [ -z "$B" ] && [ -z "$C" ]
&& [ -z "$F" ] && [ -z "$U" ] && [ -z "$S" ] && [ -z "$E" ] && [ -z "$INITRD" ] ) ||
( [ -n "$X" ] && [ -n "$E" ] && [ -z "$K" ] && [ -z "$G" ] && [ -z "$B" ]
&& [ -z "$C" ] && [ -z "$F" ] && [ -z "$U" ] && [ -z "$S" ] && [ -z "$INITRD" ] ) ||
( [ -n "$K" ] && [ -n "$INITRD" ] && [ -n "$G" ] && [ -n "$B" ] &&
[ -n "$C" ] && [ -n "$F" ] && [ -n "$U" ] && [ -n "$S" ] && [ -z "$E" ] &&
[ -z "$X" ] ) ||
( [ -n "$K" ] && [ -n "$INITRD" ] && [ -n "$G" ] && [ -n "$B" ] &&
[ -n "$F" ] && [ -n "$U" ] && [ -n "$S" ] && [ -z "$X" ] ) ||
( [ -n "$X" ] && [ -n "$INITRD" ] && [ -n "$G" ] && [ -n "$B" ] &&
[ -n "$C" ] && [ -n "$F" ] && [ -n "$U" ] && [ -n "$S" ] && [ -z "$E" ] ) ||
( [ -n "$X" ] && [ -n "$INITRD" ] && [ -n "$G" ] && [ -n "$B" ] &&
[ -n "$C" ] && [ -n "$F" ] && [ -n "$U" ] && [ -n "$S" ] && [ -z "$E" ] );
then

    if [ ! -z $K ]; then
        InstallMvux_DefaultFolder
        InstallMvux_Kernel
    fi

    if [ ! -z $G ]; then
        InstallMvux_Grub
    fi

    if [ ! -z $B ]; then
        InstallMvux_Bash
    fi

```

```
if [ ! -z $C ]; then
    InstallMvux_CoreUtils
fi

if [ ! -z $F ]; then
    InstallMvux_e2fsprogs
fi

if [ ! -z $U ]; then
    InstallMvux_UtilLinux
fi

if [ ! -z $S ]; then
    InstallMvux_SysUtil
fi

if [ ! -z $X ]; then
    InstallMvux_Busybox
fi

if [ ! -z $E ]; then
    install_Emacs
fi

if [ ! -z $INITRD ]; then
    InstallMvux_Initrd
fi

InstallMvux_Startup

sleep 2

clearPKG

dialog --title "MVUX_LINUX_SETUP" \
--msgbox "Installation_Completed!\n\nExit_end_Reboot_your_system_form_USB_Device" 10 60
break;
else

    dialog --title "MVUX_LINUX_SETUP" \
```

```

    —msgbox ”Attention , „you„have„selected „a„wrong
configuration„package” 10 60
    break
fi

else

dialog —title ”MVUX_LINUX_SETUP” \
—msgbox ”\n„\n„\n„Attention , „no„package„selected !!” 10 60
break
fi

```

Listing 7.15. Mvux_Install

Në veçanti ky skript ekzekuton skriptet e mëposhtëm:

- Mvux_Install_DefaultFolder
- Mvux_Install_Kernel
- Mvux_Install_Grub
- Mvux_Install_Bash
- Mvux_Install_CoreUtils
- Mvux_Install_Sysvinit
- ClearPKG
- Mvux_Install_LinuxUtil
- Mvux_Install_e2fsprogs
- InstallMvux_Busybox
- InstallMvux_Initrd
- Install_Emacs

- Mvux_Install_Startup

InstallMvux_DefaultFolder

```
#!/bin/sh
TARGET="`cat _rootdev`"
POINT_MOUNT="/mnt/hd"
PATH_PCK="SourcePackage"

cd $PATH_PCK
echo "Extraction_package_in_progress . . ."
scompatta.sh
sleep 17
echo "Extraction_completed! \n"

if [ ! -d $POINT_MOUNT ]; then
    mkdir -p $POINT_MOUNT
    chmod +x $POINT_MOUNT
fi

if mount | grep $TARGET 1> /dev/null 2> /dev/null ; then
    umount $TARGET
    sleep 2
    mount -t ext2 $TARGET $POINT_MOUNT
else
    mount -t ext2 $TARGET $POINT_MOUNT
fi

#----- Creo le cartelle di Default Krijo kartelat e Default
echo "Creating_folder_default_in_progress . . ."
cd $POINT_MOUNT
mkdir bin boot etc dev lib sbin boot/grub lib/tls tmp var usr proc
sleep 5
echo "Creating_finished! \n"
#-----

sleep 1

#----- Creo il file console Krijo file console
echo "Creating_the_console"
cd dev
```

```
mknod console c 5 1
echo "Creating ..finished ! ..\n"
sleep 1
cd .. # esco dalla cartella dev
#-----
```

Listing 7.16. InstallMvux_DefaultFolder

Mvux_Install_Kernel

```
#!/bin/sh
POINT_MOUNT="/mnt/hd"
PATH_PCK="SourcePackage/linux-2.6.23"

echo "\n-----Install Kernel-----"
#
echo "Compiling a package Kernel ongoing ... "
sleep 14
echo "Compiling ..finished ! ..\n"
#
sleep 1

#
cd $PATH_PCK/arch/i386/boot
echo "Copy of file vmlinuz"
sleep 2
cp bzImage $POINT_MOUNT/boot/vmlinuz
echo "Copy ..finished ! ..\n"
#
sleep 1

echo "-----End Kernel-----"
```

Listing 7.17. Mvux_Install_Kernel

Mvux_Install_Grub

```
#!/bin/sh
POINT_MOUNT="/mnt/hd"
PATH_PCK="SourcePackage/grub-0.97"

echo "\n==== Install Grub ===="
#
echo "Compiling a package Grub ongoing . . ."
sleep 7
echo "Compiling finished ! \n"
#
sleep 1

#
echo "Copy files :"
echo "- stage1"
echo "- stage2"
echo "- *_stage1_5"
sleep 5
cd $PATH_PCK/stage1/
cp stage1 $POINT_MOUNT/boot/grub
cd ../stage2
cp stage2 $POINT_MOUNT/boot/grub
cp *_stage1_5 $POINT_MOUNT/boot/grub
echo "Copy finished ! \n"
#
sleep 1

echo "===== End Grub ====="
```

Listing 7.18. Mvux_Install_Grub

Mvux_Install_Bash

```
#!/bin/sh
POINT_MOUNT="/mnt/hd"
PATH_PCK="SourcePackage/bash-3.2"
```

```
cp ldd.sh $POINT_MOUNT/bin

echo "===== Install -- Bash ====="
#
echo "Compiling a package Bash on going . . ."
sleep 7
echo "Compiling finished ! \n"
#
# sleep 1

#
# echo "Copy the file bash :"
cd $PATH_PCK
cp bash $POINT_MOUNT/bin
sleep 2
echo "Copy finished ! \n"
#
# sleep 1

#
# echo "Verification of the presence of libraries"
cd $POINT_MOUNT/bin
ldd.sh bash
sleep 4
echo "Check finished ! \n"
#
# sleep 1

#
# echo "Creation of symbolic link"
ln -s bash sh
sleep 2
echo "Creating finished ! \n"
#
# sleep 1

echo "===== End -- Bash ====="
```

Listing 7.19. Mvux_Install_Bash

Mvux_Install_CoreUtils

```
#!/bin/sh
POINT_MOUNT="/mnt/hd"
PATH_PCK="SourcePackage/coreutils-6.9.90"

cp ldd.sh $POINT_MOUNT/bin
cp ldd.sh $POINT_MOUNT/sbin

echo "\n===== Install CoreUtils ====="
#
# Compiling a package CoreUtils ongoing ...
sleep 7
echo "Compiling finished\n"
#
# sleep 1

#
# cd $PATH_PCK/src
echo "Copy files :"
echo "-cat, -chgrp, -chmod, -chown, -cp, -date, -dd, -df,
hostname, ln, ls, mkdir, mkfifo, mknod,
mv, rm, rmdir, stty, su, sync, uname"
sleep 5
cp cat chgrp chmod chown cp date dd df $POINT_MOUNT/bin
cp hostname ln ls mkdir mkfifo mknod $POINT_MOUNT/bin
cp mv rm rmdir stty su sync uname $POINT_MOUNT/bin
echo "Copy finished!\n"
#
# sleep 1

#
# echo "Verification of the presence of libraries"
```

```

cd .. # esco dalla cartella src
cd .. # esco dalla cartella coreutils-6.9.90
cd .. # esco dalla cartella SourcePackage
verificaldd.sh bin
sleep 4
echo "Check finished! \n"
#
sleep 1

echo "=====End--CoreUtils=====

```

Listing 7.20. Mvux_Install_CoreUtils

Mvux_Install_Sysvinit

```

#!/bin/sh
POINT_MOUNT="/mnt/hd"
PATH_PCK="SourcePackage/sysvinit-2.86"

echo "\n=====Install--Sysvinit=====
#
echo "Creating a package Sysvinit ongoing..."
sleep 7
echo "Compiling finished! \n"
#

sleep 1

#
cd $PATH_PCK/src
echo "Copy files:"
echo "-_init"
echo "-_halt"
echo "-_shutdown"
cp init halt shutdown $POINT_MOUNT/sbin
sleep 3
echo "Copy finished! \n"
#

```

```
sleep 1

#
cd $POINT_MOUNT/dev
mknod initctl p
cd .. # esco dalla cartella dev
#



sleep 1

#
cd etc
##### - File inittab in etc
echo "Creation_of_file_inittab"
echo "#-/etc/inittab-init_daemon_configuration_file
#_Default_runlevel
id:1:initdefault:
#_System_initialization
si:S:sysinit:/etc/init.d/rc_S
#_Runlevel_skripts
r0:0:wait:/etc/init.d/rc_0
r1:1:respawn:/bin/sh
r2:2:wait:/etc/init.d/rc_2
r3:3:wait:/etc/init.d/rc_3
r4:4:wait:/etc/init.d/rc_4
r5:5:wait:/etc/init.d/rc_5
r6:6:wait:/etc/init.d/rc_6
#_end_of_etc/inittab" > inittab
sleep 4
echo "Creating_finished! \n"
#####
#



sleep 1

#
cd init.d
##### - File rc in etc/init.d
echo "Creation_of_file_rc"
echo "#!/bin/sh
```

```

#/etc/init.d/rc--runlevel_change_skript
#
PATH=/sbin:/bin
SCRIPT_DIR=“/etc/rc\$1.d”
#
# Check that the rcN.d directory really exists.
if [ -d \$SCRIPT_DIR ]; then
# Execute the kill skripts first.
-----for SCRIPT in \$SCRIPT_DIR/K*; do
-----if [ -x \$SCRIPT ]; then
-----\$SCRIPT stop;
-----fi;
-----done;
# Do the Start skripts last.
-----for SCRIPT in \$SCRIPT_DIR/S*; do
-----if [ -x \$SCRIPT ]; then
-----\$SCRIPT start;
-----fi;
-----done;
fi
#end_of_“/etc/init.d/rc” > rc
sleep 4
echo ”Creating finished! \n”
#####
#
sleep 1
#
#####
# File hostname in etc/init.d
echo ”Creation of file hostname”
echo ”#!/bin/sh
#
# hostname -- set the system name to the name stored in /etc/hostname
#
PATH=/sbin:/bin:/usr/bin:/usr/sbin; export PATH
echo ” Setting hostname.”
if [ -f /etc/hostname ]; then
-----hostname \$( cat /etc/hostname )
else
-----hostname mvux-linux

```

```
fi
#
#_end_of_hostname"> hostname
sleep 4
echo "Creating_finished! \n"
#####
#
sleep 1
#
#####
#_halt _--halt _the_system
#
#_halt _--halt _the_system
#
PATH=/sbin:/bin; export PATH
echo \" Initiating system halt.\"
halt
#
#_end_of_/etc/init.d/halt" > halt
sleep 4
echo "Creating_finished! \n"
#####
#
sleep 1
#
#####
#_reboot _--reboot _the_system
#
PATH=/sbin:/bin; export PATH
echo \" Initiating system reboot.\"
reboot
#
#_end_of_/etc/init.d/reboot" > reboot
```

```

sleep 4
echo "Creating_finished !\n"
#####
chmod +x $POINT_MOUNT/etc/init.d/*
cd $POINT_MOUNT/etc
#



sleep 1

#
echo "Creation_of_folders :"
echo "-rc0.d"
echo "-rc1.d"
echo "-rc2.d"
echo "-rc3.d"
echo "-rc4.d"
echo "-rc5.d"
echo "-rc6.d"
echo "-rcS.d"
mkdir rc0.d rc1.d rc2.d rc3.d rc4.d rc5.d rc6.d rcS.d
sleep 4
echo "Creating_finished !\n"
#



sleep 1

#
echo "Creation_of_symbolic_link"
cd rcS.d
ln -s ../init.d/local_fs S20local_fs
ln -s ../init.d/hostname S30hostname
sleep 1
cd ../rc0.d
ln -s ../init.d/local_fs K10local_fs
ln -s ../init.d/halt K90halt
sleep 1
cd ../rc6.d
ln -s ../init.d/local_fs K10local_fs
ln -s ../init.d/reboot K90reboot
sleep 1
cd $POINT_MOUNT/sbin

```

```
ln -s halt $POINT_MOUNT/sbin/reboot
ln -s init $POINT_MOUNT/sbin/telinit
sleep 4
echo "Creating finished ! \n"
#
sleep 1

echo "=====End Sysvinit ====="
```

Listing 7.21. Mvux_Install_Sysvinit

Mvux_Install_LinuxUtil

```
#!/bin/sh
TARGET=`cat /root/dev/`"
POINT_MOUNT=/mnt/hd"
PATH_PCK=SourcePackage/util-linux-2.12p"

cp ldd.sh $POINT_MOUNT/bin
cp ldd.sh $POINT_MOUNT/sbin

echo "\n=====Install Util-Linux ====="

#
echo "Compiling a package Util-Linux ongoing . . ."
sleep 7
echo "Compiling finished ! \n"
#
sleep 1

#
cd $PATH_PCK
echo "Copy files :"
echo "-mkfs"
echo "-fdisk"
echo "-agetty"
echo "-kill"
echo "-mount"
```

```

echo “-umount”
echo “-swapon”
echo “-dmesg”
sleep 5
cp disk-utils/mkfs $POINT_MOUNT/sbin
cp fdisk/fdisk $POINT_MOUNT/sbin
cp login-utils/agetty $POINT_MOUNT/sbin
#cp login-utils/login $POINT_MOUNT/bin
cp misc-utils/kill $POINT_MOUNT/bin
cp mount/mount $POINT_MOUNT/bin
cp mount/umount $POINT_MOUNT/bin
cp mount/swapon $POINT_MOUNT/sbin
cp sys-utils/dmesg $POINT_MOUNT/bin
echo “Copy finished! \n”
cd .. # esco dalla cartella util-linux
cd .. # esco dalla cartella SourcePackage
#
#
sleep 1

#
#
echo “Verification of the presence of libraries”
verificaldd.sh bin
verificaldd.sh sbin
sleep 4
echo “Check finished! \n”
#
#
sleep 1

#
#
echo “Creation of file device”
cd $POINT_MOUNT/dev
mknod null c 1 3
mknod fd0 b 2 0
mknod ram0 b 1 0
MAKDEV sd
sleep 7
echo “Creating finished! \n”
cd ..
#

```

```
sleep 1

#
cd etc
echo "Creation_of_files :"
echo "-_fstab"
echo "-_mtab"
sleep 2
touch fstab
echo "proc _/proc proc defaults 0 0
/dev/sda1 _/ ext2 defaults 1 1
" > fstab
echo -n > mtab
sleep 2
echo "Creating_finished ! \n"
#



sleep 1

#
echo "Creation_of_file_local_fs"
mkdir init.d
cd init.d
touch local_fs
echo "#!/bin/sh
#
#_local_fs_-_check_and_mount_local_filesystems
#
PATH=/sbin:/bin; export PATH
fsck -ATCp
if [ \$? -gt 1 ]; then
    echo \" Errori presenti nel filesystem! Si richiede
        intervento manuale.\"
    /bin/sh
else
    echo \" Rimontaggio di / in modalita lettura-scrittura.\"
    mount -n -o remount ,rw /
    echo -n >/etc/mtab
    mount -f -o remount ,rw /
    echo \" Montaggio del filesystem locale.\"

```

```

mount -a -t none , nosmbfs
fi
#
# end_of_local_fs" > local_fs
chmod +x local_fs
sleep 4
echo "Creating_finished! \n"
#
sleep 1

#
cd .. # esco dalla cartella init.d
cd .. # esco dalla cartella etc
cd sbin
echo "Creation_of_symbolic_link"
ln -s getty $POINT_MOUNT/sbin/getty
sleep 2
echo "Creating_finished!"
#
sleep 1

#
# Elimino gli skript che ho utilizzato
rm ldd.sh
cd .. # esco dalla cartella sbin
cd bin
rm ldd.sh
cd .. # esco dalla cartella bin
#
sleep 1

echo "=====End Util-Linux====="

```

Listing 7.22. Mvux_Install_Sysvinit

Mvux_Install_e2fsprogs

```
#!/bin/sh
POINT_MOUNT="/mnt/hd"
PATH_PCK="SourcePackage/e2fsprogs-1.40.8"

echo "\n===== Install e2fsprogs ====="

#
echo "Compiling a package e2fsprogs ongoing . . ."
sleep 7
echo "Compiling finished ! \n"
#

sleep 1

#
cd $PATH_PCK/e2fsck/
echo "Copy files :"
echo "-e2fsck"
echo "-fsck"
echo "-mke2fs"
cp e2fsck $POINT_MOUNT/sbin
sleep 2
cd ../misc
cp fsck mke2fs $POINT_MOUNT/sbin
sleep 2
echo "Copy finished ! \n"
#

sleep 1

#
cd $POINT_MOUNT/sbin
echo "Creation of symbolic links"
ln -s e2fsck fsck.ext2
ln -s mke2fs mkfs.ext2
sleep 2
echo "Creating finished !"
#

sleep 1
echo "===== End e2fsprogs ====="
```

Listing 7.23. Mvux_Install_Sysvinit

InstallMvux_Initrd

Për krijimin e Initrd është marre në konsiderate projekti: Krijimi initramfs i jashtem nga kernel

```
#!/bin/sh
POINT_MOUNT="/mnt/hd"
PATH_PCK="SourcePackage"

echo "\n==== Install -- Initrd ===="
echo "Copy of file Initrd"
cd $PATH_PCK
cp initrd.img $POINT_MOUNT/boot
cd $POINT_MOUNT
mkdir initrd
sleep 3
echo "Copy finished!\n"
echo "==== End -- Initrd ===="
```

Listing 7.24. InstallMvux_Initrd

Mvux_Install_Startup

```
#!/bin/sh
POINT_MOUNT="/mnt/hd"
TMP=/var/log/setup/tmp

if [ -r $TMP/INITRD ]; then
    INITRD=`cat $TMP/INITRD`
    if [ ! -z $INITRD ]; then

        echo "\n==== Install -- Menu.lst ===="
#
        cd $POINT_MOUNT/boot/grub
        touch menu.lst
```

```
echo "Creation_of_file_Menu.lst_ongoing . . . "
echo "default_0
timeout_10
color_cyan/red/white/blue
title _ _ _ _ _MVux
root_(hd0,0)
kernel_/boot/vmlinuz_rootdelay=10
initrd_/boot/initrd.img
boot" > menu.lst
sleep 4
echo "Creating_finished ! \n"
#
#
sleep 1
echo "=====End_Menu.lst ====="
else
echo "\n=====Install_Menu.lst ====="
#
cd $POINT.MOUNT/boot/grub
touch menu.lst
echo "Creation_of_file_Menu.lst_ongoing . . . "
echo "default_0
timeout_10
color_cyan/red/white/blue
title _ _ _ _ _MVux
root_(hd0,0)
kernel_/boot/vmlinuz_root=/dev/sda1_rootdelay=10
boot" > menu.lst
sleep 4
echo "Creating_finished ! \n"
#
#
sleep 1
echo "=====End_Menu.lst ====="
fi
else
echo "\n=====Install_Menu.lst ====="
```

```

#_____
cd $POINT_MOUNT/boot/grub
touch menu.lst
echo "Creation of file Menu.lst ongoing . . ."
echo "default 0
timeout 10
color cyan/red white/blue
title MVux
root (hd0,0)
kernel /boot/vmlinuz root=/dev/sda1 rootdelay=10
boot" > menu.lst
sleep 4
echo "Creating finished ! \n"
#_____
sleep 1
echo "_____End _____Menu. lst _____"
fi

```

Listing 7.25. Mvux_Install_Startup

ClearPKG

```

#!/bin/sh
PATH_PCK="SourcePackage"

cd $PATH_PCK
rm -r bash-3.2
rm -r coreutils-6.9.90
rm -r e2fsprogs-1.40.8
rm -r grub-0.97
rm -r linux-2.6.23
rm -r sysvinit-2.86
rm -r util-linux-2.12p
rm -r _install

```

Listing 7.26. ClearPKG

InstallMvux_Busybox

Busybox me INIT Mvux

```
#!/bin/sh
TARGET="`cat /root/dev`"
POINT_MOUNT="/mnt/hd"
PATH_PCK="SourcePackage"

# Installazione Busybox with INIT Mvux
Instalimi Busybox me INIT Mvux
# I passi per la compilazione saranno commentati
Hapat per_kompilimin_doh_tu_komentohen
# 

# Entrare nella cartella dei sorgenti ed eseguire i seguenti comandi
Hyni n_e_kartelen_e_burimeve_dhe_ndiqni_komandat

#_make_menuconfig

#_1_Dal menu principale entrare in _BUSYBOX_SETTING
Nga_menuja_kryesore_hyni_n_e_BUSYBOX_SETTING
#      - Entrare in BUILD OPTION
Hyni n_e_BUILD_OPTION
#----- Selezionare la voce : _BUILD_BUSYBOX_AS_A_STATIC_BINARY_(NO_SHARED_LIB)
Selektioni_zerat:_BUILD_BUSYBOX_AS_A_STATIC_BINARY_(NO_SHARED_LIB)
#_2_Dal menu principale entrare in _INIT_UTILITIES
Nga_menuja_kryesore_hyni_n_e_INIT_UTILITIES
#      - Deselezionare le voci:
C_selektioni_zerat:
#          - INIT
#          - POWEROFF, HALT AND REBOOT
# 3 - Dal menu principale entrare in LINUX EXT2 FS PROGS
Nga_menuja_kryesore_hyni_n_e_LINUX_EXT2_FS_PROGS
#----- Deselezionare la voce : _FSCK
C_selektioni_zerat:_FSCK
# 4 - Dal menu principale entrare in LINUX SYSTEM UTILITIES
Nga_menuja_kryesore_hyni_n_e_LINUX_SYSTEM_UTILITIES
#----- Deselezionare le voci:
C_selektioni_zerat"
#          - FSCK_MIN
#          - MKFS_MINIX
```

```

#      – MOUNT
#      – UMOUNT
# 5 – Dal menu principale entrare in NETWORK UTILITIES
Nga menuja kryesore hyni n”e_NETWORK_UTILITIES
#-----Deselezionare la voce : _HOSTNAME
C’ selektoni uzerat : _HOSTNAME

#_make

#_make_install

#N”e kete pike do t”e_krijohet nj”e kartele ”_install” q”e_doot”e p”ermbaje
kartelat _e_m”eposhtme: ”bin”, ”sbin”, ”usr”
# Supozojm”e se kemi ndjekur k”eto hapa dhe kemi n”e dispozicion
# komandat e m”eposhtme:
# – MOUNT           preso da: util-linux
# – UMONT          preso da: util-linux
# – HOSTNAME       preso da: coreutils
# – HALT           preso da: sysvinit
# – REBOOT          preso da: sysvinit
# – FSCK            preso da: e2fsprogs
# – INIT             preso da: sysvinit
# – E2FSPROGS       preso da: e2fsprogs
# – SHUTDOWN        preso da: sysvinit
# – MKE2FS          preso da: util-linux
# – ln -s e2fsck fsck.ext2     link simbolico
# – ln -s mke2fs mkfs.ext2     link simbolico

# Inseriamo questi comandi nella cartella _install/sbin
Vendosim k”eto komanda_n”e kartelen _install/sbin

if [ ! -d $POINT_MOUNT ]; then
    mkdir -p $POINT_MOUNT
    chmod +x $POINT_MOUNT
fi

if mount | grep $TARGET 1> /dev/null 2> /dev/null ; then
    umount $TARGET
    sleep 2
    mount -t ext2 $TARGET $POINT_MOUNT
else

```

```
mount -t ext2 $TARGET $POINT_MOUNT
fi

sleep 1

#_____
InstallMvux_DefaultFolder
#_____

sleep 1

#_____
InstallMvux_Kernel
#_____

sleep 1

#_____
InstallMvux_Bash
#_____

sleep 1

#_____
InstallMvux_Grub
#_____

sleep 1

#_____
InstallINIT_Busybox
#_____

sleep 1

#_____
# Traferisco la cartella bin di _install nella pen drive
Transferojm\"e_kartelen_bin\\"e _install n\"e_pen_drive
cp -r $PATH_PCK/_install/bin_$POINT_MOUNT

sleep 2
```

```

# Traferisco la cartella bin di install nella pen drive
Transferojm\"e kartelen bin t\"e install n\"e pen drive
cp -r $PATH.PCK/_install/sbin $POINT.MOUNT

sleep 2

# Traferisco la cartella usr di install nella pen drive
cp -r $PATH.PCK/_install/usr $POINT.MOUNT

cd $POINT.MOUNT/sbin
echo "\n Verification of the presence of libraries"
ldd.sh mount
ldd.sh umount
ldd.sh e2fsck
ldd.sh fsck
ldd.sh halt
ldd.sh hostname
ldd.sh init
ldd.sh mke2fs
ldd.sh reboot
ldd.sh shutdown
sleep 3
echo "Check finished!\n"
#-----#
rm ldd.sh

Questo skript richiama altri skript:
Ky skript ther\"et skript_t\"e tjere:

* InstallMvux_DefaultFolder: Vedi sopra
* InstallMvux_Kernel: Vedi sopra
* InstallMvux_Bash: Vedi sopra
* InstallMvux_Grub: Vedi sopra
* InstallINIT_Busybox: Installazione dell'INIT di Mvux in Busybox
* Nella parte finale dello skript vengono popolate le cartelle bin, sbin e usr

```

Listing 7.27. InstallMvux_Busybox

InstallINIT_Busybox

```
#!/bin/sh
TARGET="`cat /root/dev`"
POINT_MOUNT="/mnt/hd"
echo "\n===== Install --INIT --Mvux ====="
#
echo "Creation_of_device"
cd $POINT_MOUNT/dev
mknod null c 1 3
mknod fd0 b 2 0
mknod ram0 b 1 0
mknod initctl p
MAKDEV sd
sleep 7
cd .. # esco dalla cartella dev
echo "Creating_finished! \n"
#
sleep 1

#
cd etc
echo "Creation_of_files :"
echo "-fstab"
echo "-mtab"
sleep 2
touch fstab
echo "/proc /proc defaults 0 0
/dev/sda1 / ext2 defaults 1 1" > fstab
echo -n > mtab
sleep 2
echo "Creating_finished! \n"
#
sleep 1

#
##### - File inittab in etc
echo "Creation_of_file_inittab"
```

```
echo "#!/etc/init/daemon-configuration-file
#Default runlevel
id:1:initdefault:
#System initialization
si:S:sysinit:/etc/init.d/rcS
#Runlevel-skripts
r0:0:wait:/etc/init.d/rc0
r1:1:respawn:/bin/sh
r2:2:wait:/etc/init.d/rc2
r3:3:wait:/etc/init.d/rc3
r4:4:wait:/etc/init.d/rc4
r5:5:wait:/etc/init.d/rc5
r6:6:wait:/etc/init.d/rc6
#end_of_"/etc/inittab" > inittab
sleep 4
echo "Creating finished! \n"
#####
#
sleep 1

#
#
mkdir init.d
cd init.d
##### - File local_fs in etc/init.d
touch local_fs
echo "Creation_of_file_local_fs"
echo "#!/bin/sh
#
#.local_fs --check_and_mount_local_filesystems
#
PATH=/sbin:/bin:/usr/bin:/usr/sbin; export PATH

case \$1 in
start)
    echo "Checking local filesystem integrity."
    fsck -ATCp
    if [ \$? -gt 1 ]; then
        echo "Filesystem errors still exist! Manual intervention required."
        /bin/sh
```

```
    ..else
        ..echo „Remounting / as read-write.“
        ..mount -n -o remount ,rw /
        ..echo -n > /etc/mtab
        ..mount -f -o remount ,rw /
        ..echo „Mounting local filesystems.“
        ..mount -a -t nonfs ,smbfs
    ..fi
;;
stop)
..echo „Unmounting local filesystems.“
..umount -a -r
;;
*)
..echo „usage: $0 start|stop“;
;;
esac
#
#_end_of_local_fs" > local_fs
sleep 4
echo "Creating finished !\n"
#####
#
sleep 1
#
#####
# File rc in etc/init.d
echo "Creation of file rc"
echo "#!/bin/sh
#/etc/init.d/rc--runlevel-change-skript
#
PATH=/sbin:/bin
SCRIPT_DIR=“/etc/rc\$1.d“
#
#_Check that the rcN.d directory really exists .
if [ -d \$SCRIPT_DIR ]; then
#_Execute the kill-skripts first .
```

```

-----for _SCRIPT_in_\"$SCRIPT_DIR/K*; _do
-----if _[ _-x_\"$SCRIPT_]; _then
----------\"$SCRIPT_stop;
-----fi ;
-----done;
#_Do_the_Start_skripts_last .
-----for _SCRIPT_in_\"$SCRIPT_DIR/S*; _do
-----if _[ _-x_\"$SCRIPT_]; _then
----------\"$SCRIPT_start;
-----fi ;
-----done;
fi
#_end_of_\"etc/init.d/rc" > rc
sleep 4
echo "Creating_finished! \"\n"
#####
#
sleep 1

#
#####
# File hostname in etc/init.d
echo "Creation_of_file_hostname"
echo "#!/bin/sh
#
#hostname--set_the_system_name_to_the_name_stored_in_etc/hostname
#
PATH=/sbin:/bin:/usr/bin:/usr/sbin; export PATH
echo \" Setting_hostname.\"
if _[ _-f_\"etc/hostname\"]; _then
-----hostname\"$( cat _\"etc/hostname")
else
-----hostname_mvux-linux
fi
#
#_end_of_hostname"> hostname
sleep 4
echo "Creating_finished! \"\n"
#####
#

```

```
sleep 1

#
#####
##### - File halt in etc/init.d
echo "Creation_of_file_halt"
echo "#!/bin/sh"
#
#_halt --halt_the_system
#
#_PATH=/sbin:/bin;_export _PATH
echo\_\" Initiating system halt.\"
halt
#
#_end_of_/etc/init.d/halt" > halt
sleep 4
echo "Createoing_finished!_\n"
#####
#
sleep 1

#
#####
##### - File reboot in etc/init.d
echo "Creation_of_file_reboot"
echo "#!/bin/sh"
#
#_reboot --reboot_the_system
#
#_PATH=/sbin:/bin;_export _PATH
echo\_\" Initiating system reboot.\"
reboot
#
#_end_of_/etc/init.d/reboot" > reboot
sleep 4
echo "Creating_finished!_\n"
#####
chmod +x $POINT_MOUNT/etc/init.d/*
cd $POINT_MOUNT/etc
#
sleep 1
```

```

#_________________________________
echo "Creation_of_folders :"
echo "-_rc0.d"
echo "-_rc1.d"
echo "-_rc2.d"
echo "-_rc3.d"
echo "-_rc4.d"
echo "-_rc5.d"
echo "-_rc6.d"
echo "-_rcS.d"
mkdir rc0.d rc1.d rc2.d rc3.d rc4.d rc5.d rc6.d rcS.d
sleep 4
echo "Creating_finished !_\n"
#_________________________________

sleep 1

#_________________________________
echo "Creation_of_symbolic_link"
cd rcS.d
ln -s .. / init .d / local_fs S20local_fs
ln -s .. / init .d / hostname S30hostname
sleep 1
cd .. / rc0.d
ln -s .. / init .d / local_fs K10local_fs
ln -s .. / init .d / halt K90halt
sleep 1
cd .. / rc6.d
ln -s .. / init .d / local_fs K10local_fs
ln -s .. / init .d / reboot K90reboot
sleep 1
cd $POINT_MOUNT / sbin
ln -s halt $POINT_MOUNT / sbin / reboot
ln -s init $POINT_MOUNT / sbin / telinit
sleep 4
echo "Creating_finished !_\n"
#_________________________________

sleep 1

```

```
echo "=====End--INIT.Mvux===="
```

Listing 7.28. InstallINIT.Busybox

Install_Emacs

Emacs është një editor që mund të instalohet me konfigurimet e mëposhtme:

- Emacs me ”Busybox with INITMvux”
- Emacs me ”Kernel, Grub, Bash, CoreUtils, Utillinux, Sysvinit, E2fprog”

```
#!/bin/sh
POINT_MOUNT=/mnt/hd"
PATH_PCK=SourcePackage"

echo "\n=====Install--Emacs===="
#
echo "Emacs_package_installation_in_progress..."
cd $PATH_PCK
scmpattadeb emacs
sleep 3
cd emacs

cd bin
cp run-parts $POINT_MOUNT/bin
cp tempfile $POINT_MOUNT/bin
cp which $POINT_MOUNT/bin

cd ..
# esco dalla cartella bin

cd etc
cp -r calendar $POINT_MOUNT/etc
cp -r cron.daily $POINT_MOUNT/etc
cp -r emacs $POINT_MOUNT/etc
cp -r emacs22 $POINT_MOUNT/etc
cp -r terminfo $POINT_MOUNT/etc

cd ..
# esco dalla cartella etc
```

```
cd lib
cp -r terminfo $POINT_MOUNT/lib
cp libncurses.so.5 $POINT_MOUNT/lib
cp libncurses.so.5.6 $POINT_MOUNT/lib
cp libtic.so.5 $POINT_MOUNT/lib
cp libtic.so.5.6 $POINT_MOUNT/lib

cd .. # esco dalla cartella lib

cd sbin
cp installkernel $POINT_MOUNT/sbin

cd .. # esco dalla cartella sbin

cd usr
cp -r bin $POINT_MOUNT/usr
cp -r games $POINT_MOUNT/usr
cp -r lib $POINT_MOUNT/usr
cp -r sbin $POINT_MOUNT/usr
cp -r share $POINT_MOUNT/usr

cd .. # esco dalla cartella usr

cd var
cp -r games $POINT_MOUNT/var
cp -r lib $POINT_MOUNT/var

cd .. # esco dalla cartella var
sleep 17
echo "Installation complete! \n"
#
# sleep 1
#
cd .. # esco dalla cartella emacs
rm -r emacs
cd .. # esco dalla cartella SourcePackage
echo "Verification of the presence of libraries"
cp ldd.sh $POINT_MOUNT/usr/bin
```

```

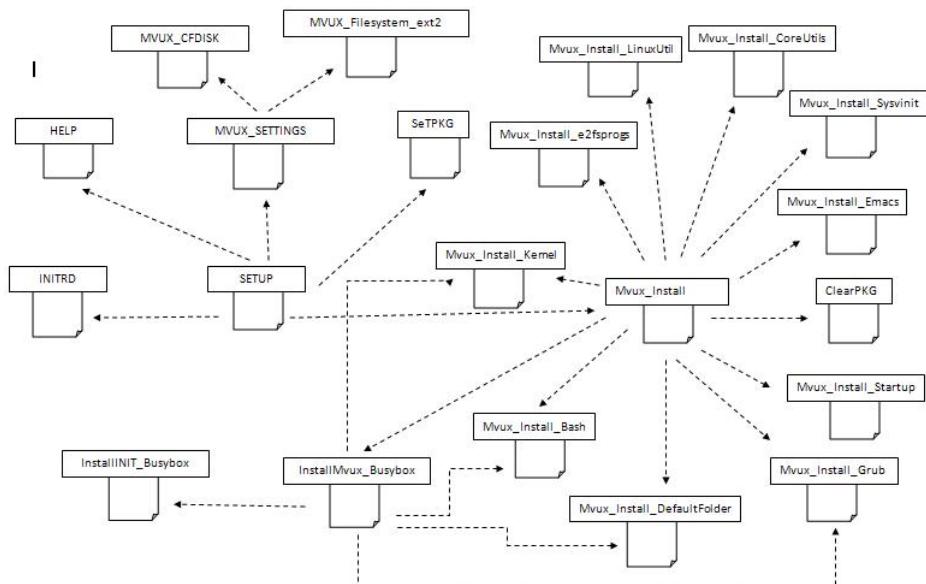
cd $POINT.MOUNT/usr/bin
ldd.sh emacs22-nox
sleep 2
rm ldd.sh
echo "Check finished! \n"
#
sleep 1

echo "=====End--Emacs===="

```

Listing 7.29. Install_Emacs

Skema logjike e instaluesit të skriptit



Startup

Për të startuar instaluesin:

- Hyn në terminal si root

```
# shtypni komandën e mëposhtme $ su
```

- Pozicionohuni në arkivin SETUP_MVUX
- Jepni komandën e mëposhtme duke u siguruar që pendrive është montuar: \$. setup

Referenca

- <http://www.slackware.com/>
- <http://www.busybox.net/>
- <http://buildroot.uclibc.org/>

7.5 Rekuperim të dhenash nga sisteme operative që nuk startojnë

7.5.1 Objektivi

Objektivi është të krijosh, duke u nisur nga distribucioni MVux, një sistem operativ me nderfaqe tekstuale, që startohet nga pendrive USB, i cili ben te mundur rekuperimin e të dhenave, ose ben të mundur kthimin e të dhenave të sistemit sic ishin me pare, në rastin kur sistemi operativ Linux ose Windows nuk starton.

Nonostante siano già presenti numerose distribuzioni Linux (sia su LiveCD sia su LiveUSB) che offrono le stesse funzionalità, questo progetto si prefigge di mostrare come sia possibile creare una distribuzione simile a partire da zero. Le altre distribuzioni, infatti, sono disponibili per il download, ma l'utente non ha nessuna conoscenza del modo in cui esse sono state create né di quello che accade durante l'avvio del sistema. Al termine della lettura, invece, l'utente sarà in grado di comprendere i vari passi effettuati per la realizzazione e il funzionamento dei principali programmi che permettono l'avvio del sistema operativo Linux. Megjithese tashme

ekzistojnë shume distribucione Linux (si në LiveCD dhe në LiveUSB), që ofrojnë të njejtat funksione, ky projekt ka si synim të tregojë se është e mundur të krijohet një distribucion i ngashem duke u nisur nga zero. Distribucionet e tjera, në fakt, janë të disponueshme për download, por përdoruesi nuk ka asnje njoħuri sesi ato janë krijuar dhe as se çfare ndodh gjatë nisjes se sistemit. Në përfundim të leximit, lexuesi do të jete në gjendje të kuptojë hapat e ndryshem të bera për krijimin dhe funksionimin e programeve kryesore qe lejojnë nisjen e sistemit operativ Linux.

Ideja baze Si e' pensato pertanto di fornire all'utente un sistema Linux avviabile da pendrive usb che: U mendua t'i jepet përdoruesit një sistem Linux, që starton nga pendrive dhe ben të mundur:

- njoħjen dhe montimin automatik të particioneve windows që gjenden në hard disk
- rekuperimin automatik për informacionet e rendesishme si adresat e regjistratura në web nga browser dhe emailet e memorizuara nga programet kryesore të postes elektronike
- që përdoruesi të zgjedhe individualisht cilat të dhenat deshiron të rekuperojë nëpërmjet një file manager
- të tentohet i sistemit operativ që nuk funksionon nëpërmjet kopjimit të disa arkivave të nevojshem në gjatë startimit (kujdo i ka ndodhur të mos startojë dot Windows XP për shkak të "Missing Ntldr" ose prej humbjes se file "boot.ini")

7.5.2 Krijimi i filesystem

Buildroot

Për krijimin e distribucionit jemi nisur nga Buildroot. Paketa Buildroot është një bashkesi me makefile dhe patch, e cila ben të mundur të krijosh lethesisht një filesystem Linux baze, duke përdorur librarinë uClibc. Behet fjale për një librari C për zhvillimin e një sistemi Linux embedded. Ajo është shume me e vogel se GNU C Library tradicionale, por pothuajse të gjitha aplikimet të suportuara nga glibc (GNU C Library) punojnë në mënyrë përfekte edhe me uClibc.

U shkarkua një version Buildroot nga faqja zyrtare. Cdo dite publikohet një version, pra data mund të jetë çfarendo. Paketa u kopjua dhe dekomprimua ne arkivin /usr/src në të cilën zakonisht mbajme kodet e programeve të sistemit.

```
cp buildroot-*.tar.bz2 /usr/src/
cd /usr/src
tar xvf buildroot-*.tar.bz2
```

Pastaj kalojmë tek konfigurimi dhe kompilimi i Buildroot.

```
cd buildroot
make menuconfig
```

Në kompilim është e rendesishme të aktivizojmë disa opsione dhe të përfshijme disa instrumenta si Busybox, Bash dhe Udev.

Busybox bashkon në një file të vetëm programe utility me permasa të reduktuara. Kjo zëvendëson programet që zakonisht gjenden ne paketat GNU fileutils, shellutils, etj. Programet në Busybox per gjithesht kanë me pak opsione në krahasim me GNU origjinale, sidqofte opzionet e përfshira i jepin përdoruesit funksionalitetet e

pritura dhe sjellja e tyre shume e ngjashme me ekivalenetet GNU. Busybox ofron një ambient të plote për cdo sistem të vogel ose embedded dhe është shkruar me objektivin për të optimizuar permasat dhe për të kursyer burimet. Është ekstremisht modular prandaj është e mundur të përfshihen ose të perjashtohen komanda(ose karakteristika) gjatë komplimit. Kjo e ben të thjeshte per personalizimin e sistemeve embedded. Për të krijuar një sistem funksional, është e mjaftueshme të shtohen nyjet device ne/dev, pak arkiva konfigurimesh dhe një kernel Linux.

Bash, si akronim i Bourne Again SHell, është një shell tekstual i projektit GNU i përdorur në sistemet operativ Unix dhe Unix-like. Behet fjale për një interpretues komandash, që i krijon mundësi përdoruesit të komunikojë me sistemin operativ nëpërmjet një serie funksionesh të paracaktuara, ose të ekzekutojë programe. Perveç kësaj, bash ve në dispozicion një gjuhe të thjesht për scripting, me të cilën mund të zgjidhen probleme me komplekse.

Udev është menaxheri i pajisjeve për sistemet Linux, dhe është ai që ”popullon” arkiven /dev gjatë startup të sistemit, e perditeson gjatë futjes se pajisjeve të jashtme dhe ekzekuton veprimet në hapesiren e përdoruesit, të tilla si ngarkimi i firmware, startimin e daemon për menaxhimin, etj. Duke filluar nga versioni 0.070 është në gjendje të beje te njejtat gjera si *hotplug* për kernelin 2.4, por është shume me i shpejte dhe me i lehte (eshte shkruar në C). Për me teper udev është në gjendje të krijojë dinamikisht arkivat device (qe i gjejmë në /dev) për çdo pajisje që njihet nga sistemi. Udev mbështetet vetëm në sysfs: Ky fakt sjell avantazhin e madh të shfrytezimit të plote të nderfaqesit të ri dhe të fuqishem të kernel 2.6 (pra sysfs) për komunikimin midis programeve në hapesiren e përdoruesit dhe driver të pajisjeve në hapesiren kernel, duke përfshire edhe funksione të reja dhe kontroll me të mirë mbi drivers.

Keto janë opzionet e aktivizuara në menuconfig të buildroot:

```
Toolchain ---> Enable large file (files > 2 GB) support ?  
GCC compiler Version (assicurarsi che la versione in uso sia la stessa di quella utilizzata dalla  
distribuzione Linux installata sul pc,
```

verificabile tramite il comando "gcc -v". Se la versione dovesse essere diversa allora scegliere quella corrispondente).

```
Package Selection for the target ---> BusyBox
Run BusyBox's own full installation
bash (se non appare, deselectare la voce "Hide applications that are provided by busybox")
Hardware handling / blockdevices and filesystem maintenance ---> udev
```

Versioni i Buildroot i p\"er dorur \"esht\"e i dates 14/02/2009. N\"e versionet e mevonshme ndryshoi politika e busybox dhe buildroot, ku udev u rishkrua me nj\"e applet t\"e p\"erfshire n\"e busybox.

Pas p\"erfundimit t\"e konfigurimit \"esht\"e e domosdoshme t\"e ruajme ndryshimet me "Save an Alternate Configuration File", duke konfirmuar emrin e file .config. Pasi dalim nga menu dhe rikthehem i n\"e shell, mund t\"e vazhdojm\"e me komplimin n\"ep\"er jet komand \"es make (komplimi mund te zgjase p\"er nj\"e kohe t\"e gjate, n\"e varesi t\"e shpejtësise se procesorit). N\"e p\"erfundim t\"e komplimit gjeni filesystem root rootfs.i686.ext2 t\"e kompiluar n\"e arkivin binaries/uclibc. Keto duhet te kopjoohen n\"e /usr/src/ dhe montuar n\"e nj\"e arkiv t\"e krijuar enkas.

```
cd /usr/src
cp buildroot/binaries/uclibc/rootfs.i686.ext2 .
mkdir brmount          #la cartella pu\oa avere un nome qualsiasi
mount -o loop rootfs.i686.ext2 brmount
```

Nese zhvendosemi në arkivin brmount gjejmë një strukture arkivash që perbejne filesystem Linux. Shtë e rendesishme të vihet re mungesa e arkivit boot. Ky duhet të krijohet manualisht, se bashku me nenarkivin grub, dhe do të përbaje file të nevojshme për startimin e sistemit dhe imazhin e kernel.

```
cd /usr/src;brmount
mkdir boot
cd boot
mkdir grub
```

Rregullat e udev

Udev bazon funksionimin e saj në disa rregulla me ane të te cilave njihen pajisjet dhe kryhen veprime të caktuara mbi to. Nëpërmjet ketyre rregullave mund të

përcaktohen emërtime fikse për pajisje të caktuara (pavaresisht nga porta e përdorur për lidhjen e pajisjes). Për me teper, eshte e mundur të thirret një program/skript i caktuari menjëherë sa po pajisja njihet nga sistemi. Do distribucion Linux ka një grup të "rregullave" per, të cilat sigurojnë një skeme për emrat dhe një seri "veprimesh" default. Perdoruesi normalisht nuk ka nevojë të modifikojë ose të integrojë këto rregulla, por shpesh mund të jete e nevojshme. Raste tipike të përdorimit:

- caktohet një emër sipas preferences për një pajisje të jashtme, për shembull /dev/pendrive, /dev/webcam
- startohet automatikisht automatikisht një program, p.sh. /usr/local/bin/- download_pictures
- shtohet automatikisht një disk i jashtem në një nensistem raid

Cdo rregull është i perbere nga një pjese *match* dhe nga një pjese tjetër *assignments*. Kur një pajisje njihet ose hiqet nga sistemi, udev kalon gjithe rregullat për të gjetur ato që i korrespondojnë pajisjes dhe ekzekuton pjesen *assignments*. E gjitha kjo behet duke krahasuar ose duke caktuar vlera për ndryshore të veçanta. Eeshte e nevojshme të shtohen në arkivin /etc/udev/rules.d/ rregullat e mëposhtme të marra nga distribucioni Linux i instaluar në kompjuter (ne këtë rast Ubuntu):

```
* 05-options.rules  
* 20-name.rules  
* 30-cdrom\_id.rules  
* 40-basic-permissions.rules  
* 40-fuse.rules  
* 40-permissions.rules  
* 45-fuse.rules  
* 50-udev.rules
```

```
* 60-persistent-input.rules  
* 60-persistent-storage.rules  
* 60-symlinks.rules  
* 65-id-type.rules  
* 70-persistent-cd.rules  
* 75-cd-aliases-generator.rules  
* 80-program.rules  
* 85-alsa.rules  
* 85-hwclock.rules  
* 90-modprobe.rules  
* 95-ude late.rules
```

Una funzione importante di alcune di queste regole e' quella di creare nella directory /dev/disk/by-label dei file relativi alle partizioni dell'hard disk che hanno come nome l'etichetta delle partizioni stesse. Questi file saranno necessari al montaggio automatico, realizzato tramite uno skript, di tali partizioni. La directory by-label, così come le altre by-id, by-path, by-uuid devono essere create manualmente e vengono popolate da udev nella fase di avvio del sistema. Nje funksion i rendesishem i disa prej ketyre rregullave është që krijon në /dev/disk/by-label arkiva për partitionet e hard diskut, të cilat kanë si emërtim etiketën e particioneve. Keto file do të jene të nevojshem për montimin automatik, realizuar me skript, te particioneve. Arkivi **by-label**, ashtu si të tjerat **by-id**, **by-path**, **by-uuid** duhet të krijohen manualisht dhe popullohen nga udev në fazen e startimit të sistemit.

```
cd /dev  
mkdir disk  
cd disk  
mkdir by-id by-label by-path by-uuid
```

Tashme i gjithe filesystem mund të kopjohet në pendrive. Më parë pendrive duhet të formatohet me një filesystem Linux. Pasi të lidher pendrive, mjafton te ekzekutohet komanda:

```
cfdisk /dev/sda      # sda deve essere il file di device corrispondente alla  
# pendrive, pu\oa anche essere sdb, sdc, ecc.
```

te krijohet një particion primar i tipit Linux filesystem ext2, të konfirmohet veprimi dhe të dalim nga cfdisk. Te gjitha këto veprime duhet të kryhen nga super-përdoruesi, sepse në të kundërt ”nuk je dikush që mund të fshije pendrive”.

Dopodiche' e' necessario creare il filesystem sulla pendrive tramite il comando Më pas duhet krijuar filesystem në pendrive duke përdorur komandën

```
mkfs.ext2 /dev/sda1
```

Në disa raste mund të nevojitet të krijohet filesystem duke modifikuar permusat e *inode*, kjo nëpërmjet komandës

```
mkfs.ext2 -I 128 /dev/sda1
```

Ora e' possibile montare la pendrive in un qualsiasi mount point e copiare al suo interno il filesystem precedentemente creato: Tani është e mundur të montohet Pendrive në një pike montimi çfaredo dhe të kopjohet në të filesystem i krijuar me pare:

```
mount /dev/sda1 /media/disk  
cp -r /usr/src/brmount/* /media/disk/
```

Më tej, është e nevojshme të krijohen në /dev të pendrive arkivat e mëposhtme speciale:

```
mknod null c 1 3  
mknod zero c 1 5
```

```
mknod console c 5 1  
mknod tty1 c 4 1
```

Komanda mknod sherben për të krijuar arkiva të veçanta me blloqe ose me karaktere. Nje arkiv i veçante perbehet nga një treshe (boolean, integer, integer) e ruajtur në filesystem. Për këtë arsyе një file i tipit të veçante nuk zë pothuajse aspak hapesire në disk dhe përdoret vetëm për të komunikuar me sistemin operativ dhe jo për të ruajtur të dhena. Shpesh arkiva të veçante i referohen një pajisje hardware (disk, shirit, tty, printer) ose device të sistemit operativ (/dev/null, /dev/random).

Startimi i sistemit të krijuar behet nëpërmjet komandës init të Busybox. Kjo komande nuk suporton runlevel, që zakonisht përdoren kur përdoret sysvinit. Nga file /etc/inittab ekzekutohen skriptet rc që ndodhen në /etc/init.d/RCS dhe kryhen veprimet e tjera per startimin e sistemit, midis tyre edhe montimi i filesystem.

Duhet të bejme të mundur që filesystem të montohet në modalitet *rw*, prandaj është e nevojshme të shtohet në file inittab, menjëherë pas rreshtit null::sysinit:/bin/mount -o remount,rw / rreshti i mëposhtëm:

```
null::sysinit:/bin/mount -n -o remount,rw -t ext2      /dev/root      /
```

7.5.3 Rikompilimi i kernel

Nje pjese themelore është rikompilimi i kernel për të shtuar modulet e nevojshme për njohjen e pajisjeve. Duhet shkarkuar versioni 2.6.23 i kernel (i cili ka edhe një file konfigurimi .config funksional) nga faqja zyrtare. Paketa duhet të kopjohet dhe të dekomprimohet në /usr/src/.

```
cp linux-2.6.23.tar.gz /usr/src/  
cd /usr/src  
tar xvf linux-2.6.23.tar.gz
```

Në arkivin e dekomprimuar duhet kopjuar file .config, shkarkuar nga
<http://www.di.uniba.it/~hixos/mvuxdown/config.2.6.23>

```
cp config.2.6.23 /usr/src/linux-2.6.23/.config
```

Në vijim, me komandën

```
make menuconfig
```

kalohet në konfigurimin e kernel. Duhet të përfshihen si built-in ose si module:

```
Device Drivers ---> ATA/ATAPI/NFM/RLL support ---> <*> Include IDE/ATAPI CDROM support
    SCSI device support ---> <*> SCSI CDROM support
    <*> MMC/SD card support

File systems ---> <*> Kernel automonter support
    <*> Kernel automonter version 4 support
    <*> Filesystem in userspace support

CD-ROM/DVD Filesystems ---> <*> ISO 9660 CDROM file system support
    [*] Microsoft Joliet CDROM extensions
    [*] Transparent decompression extension
    <*> UDF file system support

DOS/FAT/NT Filesystem ---> <*> MSDOS fs support
    <*> VFAT (Windows-95) fs support
    (437) Default codepage for FAT
    (iso8859-1) Default iocharset for FAT
    <*> NTFS file system support
    [*] NTFS debugging support
    [*] NTFS write support

Native Language Support ---> <*> Codepage 437 (United States, Canada)
    <*> Codepage 850 (Europe)
    <*> ASCII (United States)
    <*> NLS ISO 8859-1 (Latin 1; Western European Languages)
    <*> NLS UTF-8
```

Në përfundim të konfigurimit, pasi kemi ruajtur zgjedhjet tona dhe pasi kemi dale nga menu, mund të vazhdojmë me komandën

`make`

Kujdes! Ekziston mundësia që në përfundim të kompilimit të shfaqen mesazhe gabimit, që tregojnë se veprimet nuk përfunduan me sukses. Keto gabime shaktohen nga versioni i kompilatorit C të përdorur nga distribucioni Linux i instaluar në kompjuter. Ubuntu, për shembull, përdor versionin 4.3. Me këtë version, kompilimi nuk mund të përfundojë sic duhet. Mund të verifikoni versionin e kompilatorit `gcc -v`. Për të parandaluar gabime të tillë mjafton të fshini link simbolik të quajtur `quajtur` `gcc` në arkivin `/usr/bin` (qe i referohet `gcc-4.3`) dhe të rikrijoni duke e drejtuar tek `gcc-3.4`. Me këtë version të `gcc`, kompilimi përfundon me sukses.

```
cd /usr/bin  
rm gcc  
ln -s gcc-3.4 gcc
```

Në vijim mund të rivendosni gjendjen e meparshme, duke perseritur proceduren dhe duke rikrijuar linkun simbolik me `gcc-4.3`. Kompilimi i kernel ka si output një file me emrin `bzImage` në `/usr/src/linux-2.6.23/arch/i386/boot`. Ky duhet të kopjohet, duke e riemërtuar në `vmlinuz` (me marreveshje), në arkivin `boot` që nuk është i pranishem në filesystem të krijuar me `Buildroot`. Modulet e zgjedhura si built-in janë përfshire në kernel, të tjerat, pra ato që janë parashikuar si module, duhet të instalohen veç duke përdorur komandën

```
make modules\_install
```

Në përfundim, në arkivin `/lib/modules` krijohet arkiva sipas versionit të kernel (2.6.23 në këtë rast). Aty ndodhet një file me emrin `kernel`, e cila përmban modulet e kompiluar që mund të aktivohen me pas me komandën `modprobe emri_modulit`. Më pas arkivi duhet kopjuar në pendrive duke respektuar path.

Arkivi boot

Arkivi boot përfshin bootloader GRUB dhe imazhin e kernel. GRUB (Grand Unified bootloader) ben të mundur nisjen e sistemit operativ duke ngarkuar imazhin e kernel në memorie dhe duke e ekzekutuar ate. Duhet të shkarkohet versioni 0.97 i Grub dhe duhet dekomprimuar në /usr/src

```
cp grub-0.97.tar.gz /usr/src/
cd /usr/src
tar xvf grub-0.97.tar.gz
```

Më pas duhet të konfiguroni dhe kompiloni paketën.

```
cd grub-0.97
./configure
make
```

Në përfundim të kompilimit duhet të kopjohen arkivat stage1 dhe stage2 që i gjejmë në vendet respektive në dosjet e tyre perkatese, te gjitha arkivat që mbarojnë _stage1 _5 nga dosja stage2 dhe programin grub nga dosja grub.

```
cd /usr/src/grub-0.97
cp stage1/stage1 /usr/src/brmount/boot/grub/
cp stage2/stage2 /usr/src/brmount//boot/grub/
cp grub/grub /usr/src/brmount//boot/grub/
cp stage2/*\_stage1\_5 /usr/src/brmount//boot/grub
```

Tanime është e mundur të kopjohen në dosjen boot imazhin e kernel të krijuar me pare, të riemërtohet në vmlinuz

```
cp /usr/src/linux-2.6.23/arch/i386/boot/bzImage /usr/src/brmount/boot/vmlinuz
```

Filesystem është gati i plote. Ai është i perbere nga *bin boot dev etc home lib mnt opt proc root sbin sys tmp usr var*. Në /boot/grub ende mungon një file i nevojshem

për bootloader dhe për përdoruesin për startimin e sistemit: ky është menu.lst. Ai përmban listen (ne rastin kur janë me shume se një) e sistemeve operative, pozicionin ku ndodhet imazhi i kernel (vmlinuz) dhe pajisjen ku do kërkohet filesystem.

```
cd /boot/grub  
gedit menu.lst
```

Më poshtë është paraqitur përbajtja e menu.lst

```
default 0  
title MVux – kernel 2.6.23  
root (hd0,0)  
kernel /boot/vmlinuz root=/dev/sda1 rootdelay=10  
boot
```

Listing 7.30. menu.lst

7.5.4 Ntfs-3g dhe Fuse

Në versionin 2.4 të Linux kernel parashikohej mundësia për të montuar një particion me filesystem NTFS si read-only: kjo ishte sigurisht një kufizim per ata që ndajne Windows dhe Linux në një sistem *dual boot* dhe ndoshta kishte nevojë për të ruajtur arkiva të shkarkuar me Linux në një particion ose disk të formatuar me NTFS. Në kernel 2.6 është futur suporti për montimin e NTFS në modalitet *rw*, megjithese me disa kufizime të rendesishme. Në fakt, ky modul i ri kernel mund të modifikojë një file ekzistues (por pa ndryshuar permusat) dhe nuk mund të krijojë arkiva, ashtu sic nuk mund te krijojë dosje të reja.

NTFS-3G është një driver NTFS, me kod të hapur dhe falas për Linux, Mac OS X, FreeBSD, NetBSD, BeOS dhe Haiku, leshuar me një GNU General Public License, me të cilin mund të aksesohen në modalitet *rw* particione të formatuara me

filesystem NTFS. Siç dihet NTFS është një filesystem i mbyllur. NTFS-3G ofron një mënyrë efektive dhe të thjeshte për të hyre në filesystem të sistemeve operative Windows XP, Windows Server 2003, Windows 2000 dhe Windows Vista. Qellimi i projektit është të zhvilloje, duke vazhduar permirësimin dhe pasurimin e funksionaliteteve, suportin për filesystem NTFS për përdoruesit e GNU/Linux dhe të tjera sisteme Unix-like, duke siguruar nderveprimin me filesystem NTFS. Ndryshe nga driver të tjere NTFS (si për shembull ai që përfshihet në kernelin Linux), NTFS-3G ofron suport të plote për modalitetin *write*. Mund të krijohen, riemërtohen, levizur ose fshire arkiva të çfaredo madhesie. Ky driver funksionon në *user space*; partitionet NTFS montohen duke përdorur një *framework* për menaxhimin e filesystem FUSE. FUSE (*Filesystem in Userspace*) është një projekt *open source*, leshuar me license GPL dhe LGPL, për realizimin e një moduli për kernel Linux, që i mundeson përdoruesit të paprivilegjuar të një sistemi të krijojë një filesystem të tijint pa pasur nevojë të shkruaje kod në nivelin kernel. Ky objektiv eshte arrihet duke ekzekutuar kodin e filesystem në user-space, ndersa moduli FUSE punon vetëm si një ure për nderfaqen e kernel. FUSE tani është zyrtarisht pjese e kodit të kernel Linux duke filluar nga versioni 2.6.14 .

Para se të vazhdojmë me instalimin e NTFS-3G duhet të sigurohem se kemi instaluar paketën e plote të FUSE. Duhet shkarkuar paketa e FUSE-2.7.4.tar.gz dhe dekomprimuar në /usr/src/, si zakonisht. Pastaj mund të konfiguroni paketën duke shtuar opzionin --prefix. Kjo ben të mundur të specifikoni një path për instalimin të ndryshem nga ai i paracaktuar, që zakonisht është /usr/local/. Duhet të specifikohet ky opzion për të instaluar paketën në pendrive dhe jo në kompjuter.

```
cd fuse-2.7.4  
../configure --prefix=/media/disk/usr/local
```

Në vijim vazhdojmë me komplilimin dhe instalimin (kopjimi i arkivave në pendrive).

```
make
```

```
make install
```

Nga faqja zyrtare e ntfs-3g mund të shkarkoni versionin Stable Source Release 2009.4.4, që duhet të dekomprimohet në /usr/src/. Pasketaj mund të behet konfigurimi, komplilimi dhe instalimi.

```
cd ntfs-3g-2009.4.4
```

```
./configure --prefix=/media/disk/usr/local
```

```
make
```

```
make install
```

Nje mënyrë e shpejte për të verifikuar se komanda ntfs-3g e sapokopjuar funksionon, është të startohet programi nga pendrive. Me komandën

```
chroot /media/disk
```

ndryshohet arkiva root nga / në /media/disk (ku është montuar pendrive) dhe të gjithe komandat që thirren ekzekutohen nga pendrive. Nese komanda ntfs-3g funksionon, ateherë duhet të shfaqet dicta e tille:

```
ntfs-3g: No device is specified.
```

ntfs-3g 2009.4.4 integrated FUSE 27 - Third Generation NTFS Driver

Copyright (C) 2005-2007 Yura Pakhuchiy

Copyright (C) 2006-2009 Szabolcs Szakacsits

Copyright (C) 2007-2009 Jean-Pierre Andre

Copyright (C) 2009 Erik Larsson

Usage: ntfs-3g [-o option[,...]] <device|image_file> <mount_point>

```
Options: ro (read-only mount), remove\_hiberfile, uid=, gid=,
         umask=, fmask=, dmask=, streams\_interface=.

Please see the details in the manual (type: man ntfs-3g).
```

Example: ntfs-3g /dev/sda1 /mnt/windows

Ntfs-3g news, support and information: <http://ntfs-3g.org>

ne të kundërt duhet të kontrollojmë cilat janë libraritë e nevojshme për funksionimin normal

```
ldd /bin/ntfs-3g
```

```
linux-gate.so.1 => (0xb7fc9000)
libntfs-3g.so.54 => /lib/libntfs-3g.so.54 (0xb7f75000)
libpthread.so.0 => /lib/tls/i686/cmov/libpthread.so.0 (0xb7f5c000)
libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7df8000)
/lib/ld-linux.so.2 (0xb7fca000)
```

dhe të verifikohet nëse janë të gjitha të pranishme në pendrive: neqoftese mungon ndonje, duhet të kopjohen.

7.5.5 File manager

Për menaxhimin e të dhenave u mendua të përdoret një file manager i quajtur **Midnight Commander**. Midnight Commander është një file manager per Unix dhe sistemet Unix-like. Ky është një aplikacion GNU me nderfaqe tekstuale. Ekrani kryesor perbehet nga dy panele ku tregohet filesystem. Perdorimi është i ngjashem me ato të aplikacioneve të tjera që punon me *command line* Unix. Tastet shigjeta

ju mundeson te spostoheni në listen e arkivave, tastin Enter për të zgjedhur arkivat dhe tastet funksion kryejne veprime të tilla si fshirje, riemërtim, modifikim, kopjim arkivash, etj. Midnight Commander është në gjendje të riemërtojë një grup arkivash me një veprim të vetëm, në ndrshim nga file manager të tjere, qe ju detyrojnë të riemërtuni nga një file për cdo herë. Kjo karakteristike është veçanerisht e dobishme kur është nevoja të ndryshoni emrin e një numri të madh arkivash, në mënyrë që ata të jene në perputhje me një standard të paracaktuar.

Midnight Commander përfshin një editor teksti të quajtur mcedit. Mcedit ekzekutohet me vete dhe në mënyrë të pavarur. Ai ofron karakteristika si evidentimi i sintakses për gjuhe programimi dhe mundësine për të punuar edhe në formatin hekzadecimal. Pra paketa mc-4.6.1.tar.gz duhet të shkarkohet nga faqja zyrtare dhe të dekomprimohet në /usr/src/. Në vijim behet komplimi, konfigurimi dhe instalimi

```
cd mc-4.6.1
./configure --prefix=/media/disk/usr/local
make
make install
```

si dhe kopjohen libraritë që mungojnë

```
cp /usr/lib/libgmodule-2.0.so.0 /media/disk/usr/lib/
cp /usr/lib/libglib-2.0.so.0 /media/disk/usr/lib/
cp /lib/libslang.so.2 /media/disk/lib/
cp /lib/libnsl.so.1 /media/disk/lib/
cp /lib/tls/i686/cmov/libnsl.so.1 /media/disk/lib/tls/i686/cmov/
cp /lib/libdl.so.2 /media/disk/lib/
cp /lib/tls/i686/cmov/libdl.so.2 /media/disk/lib/tls/i686/cmov/
cp /lib/libpcre.so.3 /media/disk/lib/
cp /lib/tls/i686/cmov/libm.so.6 /media/disk/lib/tls/i686/cmov/
```

```
cp /usr/lib/libgpm.so.2 /media/disk/usr/lib/
```

7.5.6 Skripte

Montimi

Cdo particion në hard disk identifikohet nga një etiketë e caktuar nga prodhuesi ose përdoruesi. Skripti në vijim me emrin ”montimi” merr emrat e etiketave të partizioneve dhe i monton në pika që kanë të njejtin emër me etiketën. Duke eksploruar arkivin /dev/disk/by-label gjejmë pikerisht arkivat që përfaqesojnë particionet dhe kanë emra të njejta me etiketat. Skripti i merr emrat nga këto arkiva.

```
#!/bin/sh

ls -l /dev/disk/by-label | cat > /tmp/diskbylabel

cat /tmp/diskbylabel | awk '{ print $11 }' > /tmp/dispositivi_-
cut -d/ -f 3 -s /tmp/dispositivi_- > /tmp/dispositivi

cat /tmp/diskbylabel | awk '{ print $9 }' > /tmp/etichette

filelabel="/tmp/etichette"
filedevice="/tmp/dispositivi"

righe=$(wc -l $filelabel | awk '{print $1}')
righeD=$(wc -l $filedevice | awk '{print $1}')
riga=0
rigaD=0

while [ $riga -lt $righe ] && [ $rigaD -lt $righeD ]; do
    let riga+=1
    let rigaD+=1
    etichetta=$(head -$riga $filelabel | tail -1)
    dispositivo=$(head -$rigaD $filedevice | tail -1)
    # echo $etichetta
    # echo $dispositivo
    mkdir /mnt/$etichetta
    echo "mounting /dev/$dispositivo /mnt/$etichetta"
    ntfs-3g /dev/$dispositivo /mnt/$etichetta
done
```

```
rm /tmp/diskbylabel /tmp/dispositivi\ - /tmp/dispositivi /tmp/etichette
```

Listing 7.31. montaggio

Menu

Questo skript di nome ”menu” rappresenta l’interfaccia con l’utente all’avvio del sistema. Esso permette all’utente di compiere alcune azioni, tra cui anche quella di avviare la shell ed eseguire qualsiasi altra operazione (per l’utente esperto). Ky skript i quajtur ”menu” është nderfaqe me përdoruesin gjatë startimit të sistemi. Kjo i krijon mundësine përdoruesit te kryeje disa veprime, ndër të cilat startimin e shell dhe ekzekutimin e cdo veprimi tjetër (per përdoruesin ekspert).

```
#!/bin/sh

clear

echo " "
echo " _MAIN MENU_ choose an option and press ENTER"
echo " _____"
echo " 1 -- Try to recover Windows (copying system file from pendrive to disk)"
echo " 2 -- Save e-mail from Outlook Express or Windows Mail"
echo " 3 -- Save Mozilla Firefox bookmarks"
echo " 4 -- Save Internet Explorer bookmarks"
echo " 5 -- Start File Manager"
echo " 6 -- Go to the shell"
echo " 7 -- Reboot the system"
echo " 8 -- Shutdown the system"
echo " _____"
echo -n " Choice: "

read scelta

case $scelta in
1) cp /rec/systemfile/* /mnt/*; echo "-----FILES COPIED-----";
sleep 2; . /rec/menu ;;

2) cp -r /mnt/* /Users/* /AppData/Local/Microsoft/Windows\ Mail /rec /
```

```
cp -r /mnt/*/Documents\ and\ Settings/*/Impostazioni\ locali/Dati\ applicazioni/
    Identities/*/Microsoft/Outlook\ Express /rec/OutlookExpressMail
echo "-----MAIL_COPIED-----"
sleep 2
. /rec/menu ;;

3) cp -r /mnt/*/*Users*/AppData/Roaming/Mozilla/Firefox/Profiles/*.default/
    bookmarkbackups /rec/FirefoxBookmarks
cp -r /mnt/*/*Documents\ and\ Settings/*/*Dati\ applicazioni/Mozilla/Firefox/
    Profiles/*.default /rec/FirefoxBookmark
echo "-----FIREFOX_BOOKMARKS_COPIED-----"
sleep 2
. /rec/menu ;;

4) cp -r /mnt/*/*Users*/Favorites/Links /rec/IEbookmarks
cp -r /mnt/*/*Documents\ and\ Settings/*/*Preferiti /rec/IEbookmark
echo "-----INTERNET_EXPLORER_BOOKMARKS_COPIED-----"
sleep 2
. /rec/menu ;;

5) mc /mnt /
. /rec/menu ;;

6) /bin/sh ;;

7) echo "Reboot the system . . ."
reboot ;;

8) echo -n "Are you sure you want to SHUTDOWN THE SYSTEM? (type Y or n): "
read risposta

if [ $risposta == "Y" ]; then
    echo "Shutdown the system . . ."
    halt
else
    . /rec/menu
fi ;;

*) echo "\$scelta\$ is not valid . . ."; echo "Choose a valid option"; sleep 3; .
    /rec/menu ;;
```

Listing 7.32. menu

Nisja automatike e skripteve

Te dy skriptet duhet të vendosen në /rec, që është krijuar posacerisht në pendrive. Secilit skript i ndryshohen të drejtat e shkrim/leximit me komandën

`chmod 755`

qe i jep të drejten e leximit, shkrimit dhe ekzekutimit ”pronarit” të arkivit, dhe të drejten e leximit dhe ekzekutimit për grupin dhe për gjithë të tjeret. Për të garantuar se skriptet startojnë gjatë startimit të sistemit, duhet të thirren nga file .bashrc i pranishem në /root. Më tej, në arkivin /rec duhet krijuar një arkiv me emrin ”systemfile”, e cila duhet të përbaje arkivat e konfigurimit që bezne të mundur startimin e Windows. Ata mund të përdoren për të rivendosur sistemin në rast se një nga këto arkiva janë demtuar ose fshire. Arkivat janë: AUTOEXEC.BAT, boot.ini, Bootfont.bin, CONFIG.SYS, IO.SYS, MSDOS.SYS, NTDETECT.COM, ntldr

Problemet e hasura

- një nga problemet e hasura u citua me lart: filesystem vetëm në lexim. Siç u tha, për të mundesuar edhe shkrimin në pendrive u shtua një rresht në inittab.
- megjithë pranine e rregullave të udev, arkivat device në /dev/disk/by-label nuk janë krijuar. Duke parë përbajtjen e disa prej rregullave të udev u vu re se thirreshin disa programe, të cilat nuk ishin në pendrive. Prandaj ishte e nevojshme të kopjoheshin këto programe

`cd /lib/udev`

```
cp ata\_id edd\_id ide\_media path\_id scsi\_id usb\_device\_name  
usb\_id vol\_id write\_cd\_rules /media/disk/lib/udev/
```

- Gjate nisjes të sistemit nga pendrive shfaqet *prompt* i BuildRoot: duhet hyni në sistem si root dhe nuk ka nevojë të jepni fjalekalimin.
- Problemet e tjera, për shembull versioni i kompilatorit, i ndryshem për Buildroot, ose rikompilimi i kernel, janë trajtuar me sipër.

7.6 Motion Detection Linux

Motion Detection Linux është një distribucion Linux i instaluar në një usb. Behet fjale për një distribucion shume minimal, që lind nga ideja për të krijuar një sistem videosurvejimi.

7.6.1 Ideja

MDL është projektuar për të shfrytezuar një webcam të zakonshme dhe një USB për të ndertuar një sistem videosurvejimi, që ofron shumicen e sherbimeve te ofruara nga sistemet e tjera të këtij lloji. Pra, objektivi kryesor është regjistrimi i çdo levizje brenda zones se mbrojtur dhe monitorimi i imazheve edhe në distanca prej mijera kilometrash nëpërmjet ekspozimit të disa sherbimeve në internet.

Parakushtet

7.6.2 Konfigurimi i Kernel

Versioni i kernel i përdorur është 2.6.23. Në konfigurimin baze të kernel, sic pame me sipër, janë shtuar driver të kartes se rrjetit, pra r8169 duke përzgjedhur në kernel:

```
Device driver --> Network device support --> Ethernet(1000 Mbit) -->
Realtek 8169 Gigabit ethernet support
```

Ishte gjithashtu e nevojshme të shtoheshin driver të webcam që kishim në dispozicion, pra driver sn9c102. Për këtë kemi përzgjedhur në kernel:

```
Device driver --> Multimedia device --> Video capture adapters -->
v4l usb devices --> usb sn9c1xx pc camera controller support
```

7.6.3 Problem me Inittab

Pas instalimit të BusyBox në pendrive, në momentin e startimit kemi hasur një problem në nisjen e procesit init, që kishte lidhje me inittab. Në veçanti, problemi ishte për shkak të faktit se inittab i marre si referim ishte konfiguruar për procesin init të sysvinit:

```
# /etc/inittab init daemon configuration file
# Default runlevel
id:1:initdefault:
# System initialization
si:S:sysinit:/etc/init.d/rc S
# Runlevel skripts
r0:0:wait:/etc/init.d/rc 0
r1:1:respawn:/bin/sh
r2:2:wait:/etc/init.d/rc 2
r3:3:wait:/etc/init.d/rc 3
r4:4:wait:/etc/init.d/rc 4
r5:5:wait:/etc/init.d/rc 5
r6:6:wait:/etc/init.d/rc 6
# end of /etc/inittab
```

Listing 7.33. inittab

Në rastin tone, duke qene se gjatë konfigurimit të BusyBox kemi shtuar edhe applet të init, arkivi sysvinit i sapostuar nuk pranohej. U desh ta zëvendësonim me këtë në vijim:

```
# /etc/inittab init daemon configuration file
::sysinit:/etc/init.d/rc S
#:askfirst:/bin/sh
::ctrlaltdel:/sbin/reboot
::shutdown:/sbin/swapoff -a
::shutdown:/bin/umount -a -r
::restart:/sbin/init
::respawn:/sbin/getty 9600 tty1
```

Listing 7.34. sysvinit

Në detaj:

```
::sysinit:/etc/init.d/rc S
```

Menaxhon eventin e nisjes duke startuar skriptin rc me parameter S; rc nuk ben gjë tjetër vecse ekzekuton të gjithe skriptet në arkivin rcS.d

```
::ctrlaltdel:/sbin/reboot
```

menaxhon eventin ctrl+alt+canc që ristarton kompjuterin.

```
::shutdown:/sbin/swapoff -a
```

```
::shutdown:/bin/umount -a -r
```

menaxhon eventin e fikjes se kompjuterit

```
::restart:/sbin/init
```

menaxhon eventin e ristartimit të kompjuterit

```
::respawn:/sbin/getty 9600 tty1
```

Pas startimit të kompjuterit hyn në pune procesi getty që nuk ben gjë tjetër vecse starton shell tty1. Ky shell është i nevojshem për login të përdoruesve.

7.6.4 Konfigurimi i rrjetit

Pas startimit normal të kompjuterit dhe pasi kemi ngarkuar driver për karten e rrjetit nëpërmjet komandës:

```
modprobe r8169
```

u verifikua nëse karta e rrjetit u njoh sakte nëpërmjet komandës:

```
ifconfig eth0
```

krijuam file /etc/interfaces që përmban konfigurimin:

```
iface eth0 inet dhcp
```

i cili tregon se nderfaqja e rrjetit eth0 merr adresen IP nëpërmjet serverit dhcp dhe jo në mënyrë statike. Më pas tentuam të startojmë klientin dhcp për të marre adresen IP për lidhjen me rrjetin duke ekzekutuar:

```
udhcpc -i eth0
```

Problemet me Udhcpc

Në këtë pike u hasen disa probleme. Në thelb, applet kërkon dy skripte bash që nuk janë në kompjuter. Keto skripte janë functions.sh DEFAULT.SCRIPT që u gjeten me pas në rrjet dhe u vendosen perkatesisht në /etc/functions.sh /usr/share/udhcpc/default.script Pas vendosjes se ketyre dy skripte në vendin e duhur, procesi i konfigurimit të rrjetit ka përfunduar me sukses.

7.6.5 Konfigurimi i webcam

Në këtë pike kemi provuar të ngarkojmë modulin e webcam nëpërmjet komandës:

```
modprobe sn9c102
```

Webcam u njoh nga sistemi pa probleme, por nuk krijohej device video0. Zgjidhje për këtë pengese ishte krijimi i device ne mënyrë manuale duke përdorur komandën:

```
mknod video0 c 81 0
```

Në këtë pike konfigurimi i webcam përfundoi me sukses.

7.6.6 Motion

Pasi kemi përcaktuar qellimin e projektit është gjetur software i pershtatshem për nevojat tona: **Motion**. Motion është një program që monitoron sinjalin video nga një ose me shume kamera dhe është në gjendje të "kuptoje" nëse nese një pjese e rendesishme në zonen e kontrolluar ka ndryshuar. Me fjale të tjera, është në gjendje të kape levizjet. Programi është shkruar në C dhe është realizuar për sisteme Linux. Motion është një instrument i bazuar në rreshtin e komandës dhe nuk ka nderfaqes grafik. do gjë është e konfiguruar nëpërmjet command line ose nëpërmjet një seri arkivash konfigurimi. Memorizimi i levizjes mund të behet në disa menyra:

- file .jpg
- file .ppm
- sequenze video .mpeg

Disa prej karakteristikave të sherbimeve të Motion janë:

- fotografon sa herë që telekamera kap një levizje

- krijon një filmim sa herë që telekamera kap një levizje
- administrim njekohesht për disa pajisje video
- suport për *streaming* direkt të regjistrimeve video
- kapja automatike e fotove të castit me intervale të rregullta (suporton edhe cron)
- ekzekuton komanda të jashtme gjatë kapjes se një levizje (per shembull dergim SMS ose e-mail)
- memorizon eventet në MySQL ose PostgreSQL.
- konsum i ulet i CPU dhe ngarkese e vogel për memorien.

Konfigurim dhe instalim

Versioni i Motion që kemi përdorur është 3.2.11, i cili në kohen e realizimit të projektit ishte i fundit. Për të instaluar Motion ekzekutohet komanda e mëposhtme, pas dekomprimimit të kodeve perkatese:

```
cd motion-3.2.11
./configure --prefix=/path/from/root
make
make install
```

Tashme Motion është instaluar në distribucionin tone. Për të konfiguruar Motion duhet të modifikojmë /etc/motion-dist.conf. Ndryshimet e bera në arkivin e konfigurimit kanë qene komentimi i pjeseve që lidhen me konfigurimin për database MySQL ose PostgreSQL dhe regjistrimin e videove. Gjithashtu kanë ndryshuar pikat e mëposhtme: Starto Motion në modalitet daemon:

`daemon on`

Arkivi ku do ruhen imazhet

`target_dir /root/cam1`

Te lejohet të shihen imazhet nëpërmjet nderfaqes web:

`webcam_localhost off`

Te lejohet kontrolli i konfigurimeve të Motion nëpërmjet nderfaqes web:

`control_localhost off`

Tani mund të startojmë distribucionin dhe Motion me komandën e mëposhtme:

`motion -c /etc/motion-dist.conf`

Pershkrimi i funksionimit

Motion mund të kontrollohet edhe nga një browser. Për të parë *stream* mjafton të shkruani si adresë:

`http://localhost:8081`

ndersa për të kontrolluar dhe ndryshuar parametrat e Motion, nderkohe që është në funksion, mjafton të shtypni:

`http://localhost:8080`

7.6.7 Server ssh

Sherbimet web të Motion nuk jepnin mundësine për një kontroll përfekt të aplikacionit, për shembull nëse mbyllej sherbimi Motion nëpërmjet nderfaqes web ateherë mbyllej edhe nderfaqesi dhe sherbimi i largët nuk mund të riaktivizohej. Për të bërë me komod kontrollin e distribucionit, u instalua një server **ssh**, ose ndryshe u krijua mundësia e menaxhimit nga larg për shell të distribucionit tone.

Instalimi i dropbear

Duke konsideruar që sistemi është pothuajse i bazuar në busybox, është zgjedhur, si server ssh, të instalohet **dropbear** si applet shtese në busybox. Versioni i dropbear që kemi përdorur është 0.52. Për të instaluar dropbear duhen zbatuar komandat e mëposhtme, pasi të kemi dekomprimuar kodet perkatese:

```
cd dropbear-0.52  
.configure --prefix=/path/from/root  
make  
make install
```

Pasi startojmë distribucionin, startojmë edhe serverin ssh me komandën:

```
dropbear -a
```

Opsioni -a sherben për të mundesuar aksesin nga cdo host.

Ketu u has një problem: dropbear nuk lejon akses nga larg. Pas disa teste kemi gjetur zgjidhje për këtë problem, duke montuar një lloj të veçante device në arkivin /dev/pts me komandën:

```
mount -t devpts devpts /dev/pts
```

Pasi bajme këtë, kemi zgjidhur problemin dhe në vijim kemi krijuar skriptin initDropbear në /etc/init.d linku simbolik in/etc/rcS.d me komandat e sapodoku-mentuara, për nisjen automatike të serverit ssh.

7.6.8 Network Filesystem

Disavantazhi kryesor në zgjidhjen e implementimit të sistemit të video-vezhgimit të pershkruar deri tani, është natyrisht mungesa e hapesires në disk. Duke patur të bajme me një distro minimale, hapesira në disk është e limituar. Prandaj është

vendosur të ruhen imazhet e marra nga webcam në një kompjuter me një server NFS në funksion.

Pershkrimi

NFS permette a un sistema di condividere file e directory con altri attraverso una rete. Utilizzando NFS, utenti e programmi possono accedere ai file presenti su sistemi remoti come se fossero dei file locali. Alcuni dei principali benefici forniti da NFS sono: NFS i mundeson një sistemi të ndaje files dhe arkiva me të tjere nëpërmjet një rrjeti. Duke përdorur NFS, përdoruesit e programit mund të hyjne në files që fizikisht ndodhen larg, si të ishin në kompjuterin lokal. Disa prej avantazheve kryesore të NFS janë:

- workstation lokale përdorin me pak memorie, sepse të dhenat e zakonshme mund të memorizohen në një kompjuter të vetëm, dhe njekohesisht te aksesohen nga të tjeret nëpërmjet rrjetit.
- Perdoruesit nuk është nevoja të kene arkiva të ndryshem në cdo kompjuter të rrjetit. Arkivat mund të ndodhen fizikisht në serverin NFS dhe të jene ne dispozicion nëpërmjet rrjetit.
- Pajisjet e arkivimit si disqe, floppy, CD-Rom dhe Usb mund të përdoren nga kompjuterat e tjere të rrjetit. Kjo e redukton numrin e ketyre pajisjeve në total.

Instalimi dhe konfigurimi i serverit

Për instalimin e serverit NFS, ekzekutoni komandën e mëposhtme në terminal:

```
sudo apt-get install nfs-kernel-server
```

Është i mundur konfigurimi i arkivave për eksport duke i shtuar në /etc/exports. Për shembull:

```
/home *(rw,sync,no_root_squash)
```

Është e mundur të zëvendësojmë * me një format çfaredo për emrat host. Është e nevojshme që deklarimi i emrave host të behet sa me specifike për të ndaluar aksesin e sistemeve të padeshiruara në mount NFS. Për të startuar serverin NFS, mund të zbatojmë komandën e mëposhtme në terminal:

```
sudo /etc/init.d/nfs-kernel-server start
```

Konfigurimi i klientit

Perdorim komandën mount për të montuar një directory NFS të ndare nga një makine tjetër, duke shtypur ne rrjesht të komandës të ngjashme me këtë që vijon duke e ekzekutuar në terminal:

```
sudo mount esempio.nomehost.com:/home /locale/nfs
```

Pika e montimit /locale/nfs duhet të ekzistoje. Nuk duhet të ketë as file, as narkiva /locale/nfs. Nje mënyrë alternative për të montuar një ndarje NFS nga një kompjuter është duke shtuar një rreshti në /etc/fstab. Ky rresht duhet të përbaje emrin e host të serverit NFS, arkivin e eksportuar nga serveri dhe arkiven në kompjuterin lokal ku do montohet ndarja NFS. Sintaksa e per gjithshme për rreshtin në /etc/fstab është si vijon:

```
esempio.nomehost.com:/home /locale/nfs nfs rsize=8192,wszie=8192,timeo=14,intr
```

7.6.9 Arkivimi i pamjeve

Pasi u konfigurua Motion dhe serveri NFS, kemi arkivuar imazhe duke krijuar një skript që përmban 100 imazhe cdo herë dhe ruan të vetmin file në dosjen e largët të

serverit NFS. Ky skript, i quajtur backup_image.sh, është vendosur në /etc/init.d dhe është krijuar një link simbolik në /etc/rcS.d për të bërë të mundur nisjen automatike. Skripti është në vazhdim:

```
#!/bin/sh
cd /root/cam/
while true;
do
    controllo=100
    numero=0
    for FILE in ./*.jpg; do
        numero=$((numero+1))
    done;
    #echo $numero
    nome=$(date +\%Y\%m\%d\%H%Mprogetti)"_motion"
    data=$(date -R)
    if [ $numero -ge $controllo ]; then
        echo $data "Creating_backup_file_"$nome >> /root/log_motion
        tar -czf $nome.tar.gz *.jpg
        rm -r *.jpg
        mv *.tar.gz /mnt/nfs/
    fi
done
```

Listing 7.35. backup_image.sh

Bibliografia

- [1] Linux From Scratch <http://www.linuxfromscratch.org/lfs/>
- [2] the Linux Documentation Project
<http://tldp.org/>
- [3] Pocket Linux Guide
<http://tldp.org/LDP/Pocket-Linux-Guide/html/index.html>
- [4] Harvey M. Deitel,David R. Choffnes - *Sistemi operativi*
Pearson Paravia Bruno Mondad, 2005
- [5] Silberschatz, Galvin, Gagne - *Sistemi Operativi: Concetti ed esempi*
Pearson Paravia Bruno Mondad, 2006 = VII Ed.
- [6] Andrew S. Tanenbaum - *I Moderni Sistemi Operativi*
Prantice Hall/Jackson Libri
- [7] Bovet & Cesati - *Understanding the Linux Kernel (3a edizione)* OReilly
<http://book.opensourceproject.org.cn/kernel/kernel3rd/index.html>
- [8] Corriero, Cozza. *The hixosfs music approach vs common musical file management solutions*, SIGMAP 2009.
- [9] Control-Escape *The Linux (Virtual) File System*
<http://www.control-escape.com/linux/lx-filesystems.html> (u.a. Gen. 2011)
- [10] Reimer Jeremy - *From BFS to ZFS: past, present, and future of file systems* -
16/3/2008 Ars Tecnica
(u.a.dic. 2010) <http://arstechnica.com/hardware/news/2008/03/past-present-future-file-systems.ars/> (u.a.Ottobre 2010)
- [11] The “virtual filesystem” in Linux . Alessandro Rubini. 1997 Kernel Korner.
<http://www.linux.it/rubini/docs/vfs/vfs.html>.(u.a.Feb 2011)
- [12] Poccia Danilo - *Che cos’ ZFS* 12/10/07 - Danilo Poccia - Tecnologia et al.

Bibliografia

- http://blogs.sun.com/danilop/entry/che_cos_C3A8_zfs (u.a.Dicembre 2010)
- [13] M.Tim Jones - *Anatomy of the Linux file system* - 3010.2007
http://www.ibm.com/developerworks/linux/library/l-linux-filesystem/
(u.a.Febbraio 2011)
- [14] M.Tim Jones - *Anatomy of the Linux virtual file system switch* - 31.8.2009
http://www.ibm.com/developerworks/linux/library/l-virtual-filesystem-switch/
(u.a.Febbraio 2011)
- [15] M.Tim Jones - *Anatomy of the Linux kernel* - 3.6.2007
http://www.ibm.com/developerworks/linux/library/l-linux-kernel/
(u.a.Feb. 2011)

Wikipedia

- [16] Wikipedia - *File system* (agg.ottobre 2010)
http://en.wikipedia.org/wiki/File_system (u.a.gen 2011)
- [17] Wikipedia - *Mainframe* (agg.ottobre 2010)
http://it.wikipedia.org/wiki/Mainframe (u.a. Dic.2010)
- [18] Wikipedia - *Formato di File* (agg. ottobre 2010)
http://it.wikipedia.org/wiki/Formato_di_file (u.a. Nov.2010)
- [19] Wikipedia - *ZFS (file system)* (agg.Nov. 2010)
http://it.wikipedia.org/wiki/ZFS_(file_system) (u.a.Dicembre 2010)
- [20] Wikipedia - *FUSE* (agg.to 24.9.2010)
http://it.wikipedia.org/w/index.php?title=FUSE&action=history
(u.a.Gen.2011)
- [21] Wikipedia - *Comparison of file systems* (agg. Genn. 2010)
http://en.wikipedia.org/wiki/Comparison_of_file_systems (u.a.Gennaio 2010)
- [22] Wikipedia - *Virtual file system* (agg.ottobre 2010)
http://en.wikipedia.org/wiki/Virtual_file_system (u.a. gen 2011)

- [23] Wikipedia - *Qt (toolkit)* (agg. Dic. 2010)
[http://it.wikipedia.org/wiki/Qt_\(toolkit\)](http://it.wikipedia.org/wiki/Qt_(toolkit)) (u.a. Dic. 2010)
- [24] wikipedia - *ext2* (agg. 8/11/2010)
<http://it.wikipedia.org/wiki/Ext2> (u.a.Gennaio 2011)
- [25] wikipedia - *aufs* - (Novembre 2010)
<http://en.wikipedia.org/wiki/Aufs> (u.a. Dic. 2010)
- [26] Wikipedia - *SquashFS* (Ottobre 2010)
<http://it.wikipedia.org/wiki/SquashFS> (u.a. gen 2011)
- [27] Wikipedia - *cramfs* (agg. Dicembre 2010)
<http://en.wikipedia.org/wiki/Cramfs> (u.a. Gennaio 2011)
- [28] Wikipedia - *Network File System (protocol)* (agg.gen.2011)
<http://en.wikipedia.org/wiki/Cramfs> (u.a. Gennaio 2011)
- [29] Sartomiki.net - *File System* - 30/3/2010
<http://sartomiki.net/index.php/appunti/59-progetto-di-sistemi-operativi/255-file-system> (u.a.Dicembre 2010)
- [30] Andries Brouwer - *Some remarks on the Linux Kernel-The Linux VFS*
<http://www.win.tue.nl/~aeb/linux/lk/lk-8.html> (u.a.Nov 2010)
- [31] Version 0.8-3 David A Rusling (TLDP) - *The Linux kernel* ebook - ver.o.8-3
<http://tldp.org/LDP/tlk/fs/filesystem.html> (u.a.Gen.2011)
- [32] linux.ChinaUnix.net - *Sorgenti del Kernel*
<http://linux.chinaunix.net/bbs/thread-1005219-1-27.html> (u.a.Gen.2011)
- [33] Stenoit.com - *Il File System di Linux 1/2 e 2/2* 30/11/2007
<http://www.stenoit.com/it/node/53> (1/2) (u.a.Gen.2011)
<http://www.stenoit.com/it/taxonomy/term/9> (2/2) (u.a.Gen.2011)
- [34] The Linux Tutorial - *The Virtual File System (VFS)*
<http://www.linux-tutorial.info/modules.php?name=MContent&pageid=278>
(u.a.Gen.2011)

- [35] Ravi Kiran UVS - *Writing a Simple File System*
<http://thecoffeedesk.com/geocities/rkfs.html> (u.a. Dic. 2010)
- [36] Rmy Card - *Design and Implementation of the Second Extended Filesystem*
- Laboratoire MASI
<http://e2fsprogs.sourceforge.net/ext2intro.html> (u.a.Feb. 2011)
- [37] Ubuntu manuals - *mke2fs* - manpages-it_0.3.4-5_all
<http://manpages.ubuntu.com/manpages/gutsy/it/man8/mke2fs.8.html>
(Dicembre 2010)
- [38] CHARLES P. WRIGHT(+altri) *Versatility and Unix Semantics in Namespace Unification* - November 2005 ACM Transactions on Storage (TOS)
<http://www.fsl.cs.sunysb.edu/docs/unionfs-tos/unionfs.html> (u.a.Gen.2011)
- [39] Sourceforge.net - *aufs*
<http://aufs.sourceforge.net/aufs2/man.html> (u.a.Gen.2011)
- [40] Sourceforge.net - *squashfs*
<http://squashfs.sourceforge.net/>
- [41] (u.a.Gen.2011) Artemiy I. Pavlov - *SquashFS HOWTO* - tldp.org
<http://tldp.org/HOWTO/SquashFS-HOWTO/whatis.html>
www.ibiblio.org/pub/linux/docs/howto/other.../SquashFS-HOWTO.pdf
(pdf version) (u.a.Gen.2011)
- [42] OpenSourceProject.org.cn *CRAMFS*
<http://book.opensourceproject.org.cn/embedded/oreillybuildembed/opensource/belinuxsys-chp-8-sect-3.html> (Febbraio 2011)
- [43] Sourceforge.net - *cramfs* <http://cramfs.sourceforge.net/> (Febbraio 2011)
- [44] Linux Kernel Documentation *Documentation / filesystems / cramfs.txt*
<http://www.mjmwired.net/kernel/Documentation/filesystems/cramfs.txt>
(u.a.Gen.2011)
- [45] Sourceforge.net - *Linux NFS Overview, FAQ and HOWTO Documents*

- <http://nfs.sourceforge.net/> (u.a.Feb.2011)
- [46] Christopher Smith - *Linux NFS-HOWTO* 2/5/2006
<http://nfs.sourceforge.net/nfs-howto/> (u.a.Feb.2011)
- [47] Kernel. L. Torvalds. www.kernel.org. Ver. 2.26.34
- [48] David Howells - *[PATCH 31/31] IGET: Remove ige() and the read_inode() super op as being obsolete [try#3]* - 10/10/2007
<http://lkml.indiana.edu/hypermail/linux/kernel/0710.1/0812.html>
- [49] Tigran Aivazian *Linux Kernel 2.4 Internals* 7 August 2002 (29 Av 6001)
<http://moses.uklinux.net/patches/lki.html#toc3> (u.a. Febb. 2011)
- [50] [linux/Documentation/filesystems/vfs.txt](http://www.kernel.org/doc/Documentation/filesystems/vfs.txt)