UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Elettronica per l'Automazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica
(Prof. Riccardo Cassinis)

Installazione e collaudo di Linux sul PC104

Elaborato di esame di: **Dennis Baranzelli, Dario Brignani**

Consegnato il: **4 Luglio 2003**

Sommario

Questo elaborato nasce dalla stratificazione di un progetto più ampio: ottenere una postazione di controllo completa e mobile che permetta maggior autonomia del robot MARMOT dalle postazioni fisse. Analizzando tale progetto, si può identificare il seguente diagramma stratificato:

MARMOTCAM	*MARMOTMOVE*
CONTROLLORE MARMOT	
ROBOT MARMOT	

MARMOTCAM e MARMOTMOVE rappresentano funzionalità ad alto livello; MARMOTCAM permette il controllo della webcam montata sul robot, e MARMOTMOVE ne gestisce la navigazione. Si noti che il controllore deve permettere di interfacciare il robot MARMOT con i servizi di più alto livello MARMOTCAM e MARMOTMOVE. Come controllore è stato utilizzato un calcolatore embedded dotato di un sistema operativo opportunamente configurato. Il problema trattato in questo elaborato riguarda l'analisi e l'implementazione del controllore MARMOT.

1. Introduzione

In questo documento ci proponiamo di illustrare i vari aspetti relativi all'installazione e alla configurazione di un sistema embedded. Partiamo menzionando l'obbiettivo propostoci per questo progetto per poi introdurre un'analisi di progettazione. Questa documentazione è stata strutturata in quattro parti principali: Il problema affrontato, La soluzione adottata, Modalità operative, e Appendici.

Nel capitolo "Il problema affrontato" si evidenzia esplicitamente gli obbiettivi da conseguire e i vincoli imposti, le scelte di progettazione vengono invece discusse nel capitolo successivo: "La soluzione adottata". L'utente interessato semplicemente all'utilizzo degli strumenti implementati può trovare le informazioni necessarie nel capitolo "Modalità operative"; inoltre per garantire un maggior supporto e una migliore comprensione sono state aggiunte in appendice alcune indicazioni utili.

Al lettore interesserà che per facilitare la stesura di questa documentazione sono state utilizzate delle convenzioni che riteniamo opportuno descrivere nel prossimo paragrafo.

1.1. Convenzioni utilizzate

- Quando sono disponibili maggiori informazioni sull'argomento trattato in un'altra parte del testo lo si indica con: (V: numero paragrafo, titolo paragrafo).
- Nel testo si fa più volte riferimento agli Hard Disk interno ed esterno; con questi due termini si indicano rispettivamente il chipDISK da 32MB presente all'interno dell'involucro del PC104 e l'Hard Disk Caviar 22500 da 2500MB; per maggiori informazioni si rimanda all'appendice A
- Per riferirci ad un calcolatore caratterizzato da piccole dimensioni e da una forte integrazione delle varie periferiche molte volte si utilizza il termine "sistema embedded"

2. Il problema affrontato

2.1. Obbiettivo

L'obbiettivo dell'elaborato consiste nel configurare il PC104 per permetterne l'uso sul robot mobile MARMOT. Questo comporta l'installazione di un sistema operativo, in questo caso linux, che sia in grado di riconoscere le periferiche installate, unitamente ad alcuni programmi di supporto.

In particolare sono richiesti:

Periferiche:

configurazione della porta USB

configurazione della scheda di rete ethernet integrata

configurazione PCMCIA bridge

configurazione della scheda PCMCIA ethernet wireless

Programmi:

installazione demone ssh

installazione demone telnet

Sistema:

installare un kernel 2.4.20

compilare nel kernel il supporto PWC (Philips webcam) utilizzando la patch pwc_8_8

configurare strumenti che facilitino l'aggiornamento e la manutenzione.

Nel progetto non sono indicati particolari vincoli di progettazione; l'unico vincolo richiede l'utilizzo di linux come sistema operativo.

3. La soluzione adottata

Il capitolo descrive la fase di progettazione del controllore di MARMOT. La progettazione seguirà un'analisi di tipo top-down affrontando il problema a grandi linee per poi descriverne i particolari. Il top-down si compone principalmente di tre livelli di astrazione: analisi generale, implementazione e approfondimenti sull'implementazione.

3.1. Analisi generale

In questo paragrafo introduciamo un'analisi generale del problema. In particolare vengono fatte delle considerazioni valutando gli obbiettivi da raggiungere aiutandoci con la stratificazione del problema. Come già accennato, si vuole implementare il blocco CONTROLLORE MARMOT (V:Sommario). Il raggiungimento dell'obbiettivo comporta l'installazione di un sistema che sia in grado di fornire i servizi necessari ai blocchi sovrastanti garantendo una certa flessibilità nella configurazione. Per facilitare la comprensione del problema si preferisce suddividerlo in obbiettivi di minor entità, stratificando in modo da evidenziare le dipendenze.

INTERFACCIA UTENTE
STRUMENTI PER LO SVILUPPO
SISTEMA OPERATIVO
SISTEMA EMBEDDED

A ragion di logica partiamo con l'analisi del sistema embedded.

3.1.1. Sistema Embedded

Ampiamente descritto nel paragrafo componenti, nel nostro caso si tratta di un PC104 (V:6.1,APPENDICE A:Descrizione dettagliata dei componenti). Come si intuisce, appartenendo allo strato 0, è lui ad imporre i maggior vincoli, di cui il più sentito è la mancanza di una memoria di massa sufficientemente grande: si dispone di soli 32MB. Il PC104 permette di ovviare a questo problema connettendo un Hard Disk esterno, che però può essere usato solo con una alimentazione di rete. Facendo una breve analisi si comprende che è necessario distinguere quando è possibile utilizzare l'Hard Disk esterno e quando No. Se il PC104 deve funzionare in modo autonomo le risorse disponibili sono ridotte al minimo e non abbiamo alternative, ma è possibile collegare l'Hard Disk esterno quando ad esempio dobbiamo solo configurarlo. Si giunge quindi ad una stratificazione del tipo:

INTERFACCIA UTENTE	
STRUMENTI PER LO SVILUPPO	
SISTEMA OPERATIVO	SISTEMA OPERATIVO
SISTEMA EMBEDDED IN	SISTEMA EMBEDDED IN

MODALITÀ SVILUPPO	MODALITÀ OPERATIVA

Nel diagramma si distinguono due modalità: la modalità operativa prevede l'uso della batteria e del solo Hard Disk interno, la modalità sviluppo prevede l'uso dell'alimentazione di rete e dell'Hard Disk esterno. Come evidenzia il diagramma è possibile distinguere due colonne unite dallo strato interfaccia utente; d'ora in poi ci riferiremo alla prima colonna con "struttura della modalità di sviluppo" e alla seconda con "struttura della modalità operativa".

3.1.2. Sistema operativo

Questo strato deve essere disponibile in ogni caso perché garantisce la funzionalità di base del PC104.

3.1.3. Strumenti per lo sviluppo

Con questo strato si intende rappresentare tutti gli strumenti necessari a configurare il sistema, ad esempio:

sorgenti per il kernel

sorgenti per la PCMCIA

archivi dei pacchetti installati

pacchetti necessari alla compilazione

script di vario genere

strumenti per il ripristino

3.1.4. Interfaccia Utente

Questo strato rappresenta l'interfaccia dell'intera struttura con l'utente. Il boot loader e al sua configurazione sono considerati come appartenenti all'interfaccia utente.

Per realizzare il tutto si sfrutteranno l'Hard Disk esterno per implementare la struttura della modalità di sviluppo, e l'Hard Disk interno per implementare la struttura della modalità operativa. L'implementazione sull'Hard Disk esterno viene fatta secondo le normali procedure di installazione linux, mentre l'Hard Disk interno verrà configurato semplicemente copiando il necessario dall'Hard Disk esterno. È facile capire dalla struttura stratificata illustrata nel paragrafo 3.1.1 (Sistema Embedded) che sarà sufficiente copiare gli strati necessari, escludendo lo strato "STRUMENTI PER LO SVILUPPO"; d'ora in poi ci riferiremo a questa operazione con "creazione dell'immagine"

3.2. Implementazione

Nell'analisi generale si è suddiviso il problema in strati descrivendo per ognuno di essi il compito e le funzionalità. In essa si è volutamente ignorata l'implementazione che verrà affrontata in questo paragrafo descrivendo le metodologie e gli strumenti utilizzati.

3.2.1. Implementazione della “struttura della modalità di sviluppo”

Come è già stato menzionato otteniamo la struttura della modalità operativa partendo da quella di sviluppo, questo ci porta ad affrontare prima l’implementazione della struttura della modalità di sviluppo. Anche qui affrontiamo il problema descrivendolo a strati e aiutandoci con il diagramma del paragrafo 3.1.1 (Sistema Embedded).

Sistema embedded in modalità di sviluppo

Questo strato rappresenta il PC104 munito di alimentazione di rete e di Hard Disk esterno. Ovviamente si necessita anche delle normali periferiche di in/out quali la tastiera, lo schermo e il floppy correttamente installate.

Sistema operativo

La struttura ha bisogno di un sistema operativo funzionante, installato correttamente sull’Hard Disk esterno. Il vincolo di progetto ci porta ad utilizzare una delle tantissime distribuzioni linux disponibili. Anche se in questa modalità, usufruendo dell’Hard Disk esterno, abbiamo molto spazio a disposizione non dobbiamo dimenticare che anche la modalità operativa deve contenere questo strato, e che quindi deve essere il più piccolo possibile. Un’approfondita ricerca ci ha portato ad installare la mini distribuzione White Dwarf 1.2 disponibile con un kernel 2.4.18 (V:3.3.2,Installare la White Dwarf 1.2). Prima dell’installazione è stato partizionato l’Hard Disk esterno secondo le nostre esigenze: la partizione 1 è utilizzata per contenere la White Dwarf 1.2 e ne rappresenta la root (V:3.3.1,Il Partizionamento).

Strumenti per lo sviluppo

Utilizziamo questo strato per contenere tutti i servizi necessari alla modalità di sviluppo ma che possono essere esclusi dalla modalità operativa. Tali contenuti possono essere identificati come:

Sorgenti per il kernel

> Se fosse necessario ricompilare il kernel o moduli aggiuntivi non potrebbero certamente mancare i sorgenti ed eventuali patch (V:3.3.4.1,Compilare il kernel).

Sorgenti per la PCMCIA

> Volendo installare la PCMCIA bisogna disporre dei sorgenti.

Archivi dei pacchetti installati

> Un archivio di alcuni dei pacchetti installati ne permette la disinstallazione e l’installazione automatica: alcuni script necessitano di fare questa operazione per ridurre lo spazio occupato (V:3.3.6,Creare l’immagine).

Pacchetti necessari alla compilazione

> Ovviamente per poter procedere alla compilazione dobbiamo disporre degli strumenti necessari installati e pronti all’uso (V:3.3.4,Appunti sulla compilazione).

Script di vario genere

> Volendo automatizzare alcuni processi ci si serve di alcuni script: kernel_update e crea_immagine ne sono un esempio. Kernel_update permette all’utente di

sostituire l'attuale kernel con quello indicato come argomento (V:3.3.7.4,Script kernel_update). Crea_immagine provvede a creare l'immagine che contiene il necessario per la modalità operativa incaricandosi inoltre di copiare il tutto sull'Hard Disk interno e di configurare il boot loader (V:,Creare l'immagine).

Strumenti per il ripristino

Dando la possibilità all'utente di modificare il kernel può succedere che la modalità di sviluppo non si riavvii più. Per ovviare a questo problema si è fatto in modo che l'aggiornamento del kernel sia preceduto dal backup di quello attuale. Per poter ripristinare la vecchia configurazione è stata installata sull'Hard Disk esterno una Slackware 9.0, a cui si accede selezionando "Ripristino" al menu di avvio (V:3.3.3,Installazione la Slackware 9.0). La modalità ripristino permette di eseguire lo script /ripristino/ripristino che provvede alla riattivazione della copia di backup.

ATTENZIONE: Lo script kernel_update non prevede backup progressivi, ciò significa che se utilizzato due volte con lo stesso kernel, viene sovrascritto sia il kernel attuale sia quello salvato.

3.2.2. Implementazione della "struttura della modalità operativa"

Il PC104 in questa struttura dispone di un'alimentazione a batteria e del solo Hard Disk interno. Valutando le circostanze, se utilizzassimo come root il file system dell'Hard Disk interno, rischieremmo ad ogni calo di tensione della batteria di danneggiare l'intera struttura software. Per ovviare a questo problema, preferiamo ad ogni avvio caricare l'immagine della root in un RAM disk (V:3.3.5,RAM disk come file system root). Per attuare questa tecnica sull'Hard Disk interno devono essere presenti due file: vmlinuz che contiene il kernel compilato con il supporto per i RAM disk e un file image.gz contenente l'immagine della root da caricare. Se si utilizza come boot manager LILO basterà aggiungere le seguenti direttive nel file lilo.conf:

image=/..../..../kernel

initrd=/…./..../image.gz

root=/dev/ram0

Per maggiori informazioni si rimanda il lettore all'approfondimento del paragrafo 3.3.7.3 (Script lilo_update). L'utilizzo di un RAM disk comporta ovviamente l'impossibilità da parte dell'utente di modificare il contenuto della root in modo permanente: infatti, ad ogni riavvio, viene ripristinata la configurazione originale. Si fa eccezione per la directory /flash nella quale viene montato in modo automatico l'Hard Disk interno.

Directory /flash:

Come è già indicato attraverso questa directory è possibile accedere al file system dell'Hard Disk interno. Il suo contenuto è quindi modificabile in modo permanete ed è possibile utilizzarla tranquillamente come supporto di memorizzazione. Per evitare possibili anomalie diamo un tracciato del contenuto indicando i file da non cancellare:

CONTENUTO	UTILIZZO
flash/dev/	Contiene i file di dispositivo necessari all'installazione del boot loader LILO
flash/boot/vmlinuz	Kernel caricato quando si usa la modalità operativa
flash/boot/kernelsviluppo	Kernel caricato quando si usa la modalità di sviluppo.
flash/boot/boot.b	File necessario all'installazione del boot loader LILO
flash/boot/lilo.conf	File necessario all'installazione del boot loader LILO
flash/boot/image.gz	Immagine caricata nel RAM disk e utilizzata come root quando si usa la modalità operativa.

L'immagine e la configurazione del boot loader si ottiene direttamente in modalità di sviluppo avviando lo script crea_immagine.

ATTENZIONE:

Se va perso tutto il contenuto della directory /flash per ristabilire il buon funzionamento del sistema è necessario riavviare lo script /sviluppo/script/crea_immagine dalla modalità di sviluppo.

3.3. Approfondimenti sull'Implementazione

3.3.1. Il Partizionamento

Durante il processo d'installazione della White Dwarf 1.2 è possibile procedere al partizionamento:

PARTIZIONE	DIMENSIONE	CONTENUTO
1	360 MB	
2	1400 MB	
3	N/A	Rappresenta l'inizio della partizione estesa
4	N/A	Non disponibile
5	60 MB	
6	580 MB	Contiene sorgenti vari : kernel, PCMCIA, moduli

La colonna "CONTENUTO" non risulta essere completa perché saremmo costretti ad inserire concetti che verranno introdotti solo successivamente. Questa tabella viene riprese più volte per aggiornarne i contenuti. La Partizione numero 6 contiene in particolare:

sorgenti kernel nella forma compressa (file: kernel-2.4.20/linux-2.4.20.tar.bz2)

sorgenti PCMCIA nella forma compressa (file:PCMCIA/pcmcia-cs-3.2.3.tar.gz)

patch PWC per webcam Philips nella forma compressa (file:patchPWC/pwc-8.8.tar.gz)

sorgenti pacchetto openssh nella forma compressa (file:openssh/openssh-3.4p1.tar.gz)

3.3.2. Installare la White Dwarf 1.2

Introduzione

È stata scelta la White Dwarf 1.2 perché è ottimizzata per i sistemi embedded. Deriva dalla maggiore Slackware, e in quanto tale, ne eredita anche la struttura di base come la gestione dei pacchetti. Dispone di un kernel 2.4.18 ed è possibile scaricarla alla pagina www.whitedwarflinux.com; di seguito è mostrato come installarla sull'Hard Disk esterno.

Installazione

La minidistribuzione White Dwarf, ci vincola ad avere il lettore CDROM e l'Hard Disk rispettivamente nelle posizioni primary slave e primary master. Dovendo posizionare l'Hard Disk esterno come primary master non è possibile utilizzare il PC104 (il calcolatore non funziona se si scollega l'Hard Disk interno). Usufruendo di un altro PC otteniamo la configurazione adatta. Come mostra la seguente tabella delle partizioni la White Dwarf viene istallata nella prima partizione del disco esterno:

PARTIZIONE	CONTENUTO
1	File system root della White Dwarf 1.2
2	
3	Rappresenta l'inizio della partizione estesa
4	Non disponibile
5	
6	Contiene sorgenti vari : kernel, PCMCIA, moduli

Quando richiesto procediamo alla selezione dei seguenti pacchetti:

Nome pacchetto	Servizio fornito
aaa_base	Struttura base del file system
Bash	Shell bash

Bzip	bzip2, bzip2recover
Devs	File dei dispositivi contenuti in /dev
e2fsbn	Manipolazione del file system
Elflibs	Librerie ELF
Elvis	Editor di testo
Etc	Setta il contenuto della directory /etc
Fileutls	ls, chmod, chown, cp, df, ln, mkdir, mv, rm, rmdir, dd, du
Find	Comando find
Grep	Comando Grep
Hotplug	In particolare gestisce l'auto-riconoscimento dei dispositivi sulla porta Usb
Kernel	Kernel 2.4.18 con moduli
Ldso	Permette la gestione dinamica delle librerie
Less	Comando less
Lilo	Boot loader LILO
Modutils	Manipolazione dei moduli
Openssl	Librerie
Pkgtool	Manipolazione dei pacchetti
Procps	Setta il contenuto della directory /proc
sh_utils	Date, false, true, stty, echo, pwd
Shadow	Gestione password
Sysklogd	Gestione messaggi di debug
Sysvinit	Script di avvio della directory /etc/rc.d
tcp_wrappers	Aggiornamento di tcpd
Tcpip	ftp, telnet, ping, hostname, ftpd, telnetd, ifconfig, route, whois, traceoute, tcpd, inetd
Txtutils	Comandi per la gestione dei file di testo
Util	Utilità di vario genere tra cui mount e unmount
Zlib	Librerie per la compressione/decompressione
Zoneinfo	Gestione della timezone

Questa è considerata la configurazione base del sistema operativo.

Ora bisogna prestare attenzione che nel trasportare l'Hard Disk su PC104 è necessario portare l'Hard Disk da master a slave e modificare il file /etc/fstab in modo che la root venga montata sul dispositivo fisico corretto riportiamo un esempio di file fstab:

Prima della modifica:

/dev/hdb1	/	ext2	defaults	1	1
none	/proc	proc	defaults	0	0

Dopo la modifica:

/dev/hda1	/	ext2	defaults	1	1
none	/proc	proc	defaults	0	0

Il sistema cosi configurato può essere avviato utilizzando un dischetto di avvio di GRUB utilizzando la seguente procedura: al prompt di GRUB digitare “kernel (hd1,0)/boot/vmlinuz root=/dev/hdb1 ro” e premere invio, poi digitare “boot” e confermare nuovamente con invio.

3.3.3. Installare la Slackware 9.0

Si procede all’installazione della Slackware per garantire la possibilità di ripristinare la configurazione della modalità di sviluppo quando questa non si riavvia.

PARTIZIONE	CONTENUTO
1	File system root della White Dwarf 1.2
2	File system root della Slackware 9.0
3	Rappresenta l’inizio della partizione estesa
4	Non disponibile
5	Swap per la Slackware
6	Contiene sorgenti vari (kernel, PCMCIA, moduli)

Come mostra la tabella sopra esposta le partizioni interessate da questa operazione sono le partizioni 2 e 5. La partizione numero 2 è destinata a contenere il file system della root e la partizione 5 a contenere l’area di swap. Si procede alla normale installazione della distribuzione prestando attenzione ad indicare in modo corretto le partizioni da utilizzare.

3.3.4. Appunti sulla Compilazione

Il sistema operativo linux prevede fasi di compilazione per ottenere la configurazione desiderata.

Talvolta risulta necessario compilare kernel, moduli, o semplici programmi. La compilazione è possibile solo in presenza di specifici strumenti; non possono mancare i sorgenti e i seguenti pacchetti devono essere correttamente installati:

autoconf-2.53-i386-1.tgz

automake-1.6.2-i386-1.tgz

make-3.79.1-i386-1.tgz

binutils-2.11.2-i386-1.tgz

gcc-3.1-i386-1.tgz

mawk-1.3.3-i386-1.tgz

glibc-2.3.1-i386-3.tgz

kernel-headers-2.4.20-i386-5.tgz

ncurses-5.2-i386-1.tgz

È importante che siano stati installati nell'ordine sopra esposto. In questo elenco non sono menzionati i pacchetti facenti parte della configurazione base del sistema operativo (V:3.3.2,Installare la White Dwarf 1.2). Per quanto riguarda i sorgenti è possibile memorizzarli nell'apposita partizione che ipotizziamo montata nella directory /mnt (V:3.3.1,Il Partizionamento). Prima di passare al prossimo paragrafo si fa presente che l'unica modalità che possiede questi requisiti è la modalità di sviluppo.

3.3.4.1 Compilare il Kernel

La ricompilazione del kernel prevede 4 passi fondamentali a cui è necessario attenersi, dopo essersi connessi al sistema come utente root. Prima di avviare tale procedura, però, è necessario che i file sorgenti del kernel siano copiati correttamente sul disco. È possibile scaricare i file sorgenti dal sito www.kernel.org. In questo caso si fa uso del file linux-2.4.20.tar.gz che è stato decompresso nella directory kernel-2.4.20 della partizione numero 6 (V:3.3.1, Il partizionamento). La decompressione creerà la sottodirectory chiamata linux-2.4.20. Il primo passo da compiere per la compilazione del kernel consiste nell'aggiornamento (o nella creazione, se è la prima volta che lo si fa) del file di configurazione del kernel. Per eseguire tale operazione bisogna portarsi nella directory dei sorgenti (kernel-2.4.20/linux-2.4.20) e digitare la seguente istruzione:

make menuconfig

Lancia il programma che permette all'utente di personalizzare il kernel. Nel seguito spiegheremo meglio questo passo.

I successivi tre comandi possono essere digitati insieme o separatamente:

make dep

Il comando "make dep" controlla che tutti i file necessari per la compilazione siano presenti, creando poi delle dipendenze tra questi file

make clean

il comando "make clean" elimina i file temporanei che potrebbero disturbare il processo di compilazione

make bzImage

Avvia la compilazione e crea il file del kernel vero e proprio.

Al termine di tale operazione il nuovo kernel si troverà nella directory kernel-2.4-20/linux-2.4.20/arch/i386/boot con il nome bzImage. Utilizzando lo script

kernel_update è possibile aggiornare l'attuale kernel con quello appena compilato(V:3.3.7.4,Script kernel_update).

Per maggiori informazioni sulla compilazione rimandiamo il lettore alla lettura di kernel-HOWTO.

3.3.4.2 Compilare il Software PCMCIA

Il processo che permette di compilare il supporto della PCMCIA è del tutto simile alla compilazione del kernel. Bisogna scaricare il drive pcmcia-cs da www.sourceforge.net. A questo punto si suggerisce di utilizzare l'apposita partizione dei sorgenti (V:3.3.1,Il Partizionamento) e di archiviare e scompattare il file pcmcia-cs-3.2.3.tar.gz nella directory /mnt/PCMCIA. Si ottiene la directory dei sorgenti /mnt/PCMCIA/pcmcia: da questa directory devono essere richiamati in successione i tre comandi make config, make all, make install.

Dopo aver lanciato make config viene chiesto all'utente di specificare:

la directory contenente i sorgenti del kernel

la directory in cui installare i nuovi moduli della PCMCIA

Seguendo la configurazione dettata in questo paragrafo vanno introdotte le seguenti directory:

/mnt/kernel-2.4.20/linux-2.4.20

/lib/modules

L'esecuzione di make all seguita da make install costruisce ed installa i moduli del kernel ed i programmi di utilità. I moduli vengono installati dentro /lib/modules/2.4.20/pcmcia/, i programmi cardmgr e cardctl vengono messi in /sbin ed infine i file di configurazione vengono installati nella directory /etc/pcmcia.

Per una maggiore documentazione sull'argomento si rimanda alla lettura del file PCMCIA-HOWTO contenuto nell'archivio dei driver PCMCIA.

3.3.5. RAM disk come file system root

Nella fase di avvio del sistema operativo è possibile attraverso la direttiva root indicare il dispositivo che contiene il file system principale. Non viene esclusa la possibilità di indicare come supporto un RAM disk che ovviamente deve contenere un file system adeguato all'avvio del sistema. Per inizializzare il RAM disk con il file system contenuto in una immagine si utilizza la direttiva initrd, con la quale si indica il file specifico. Riportiamo un esempio di configurazione di avvio nel caso si utilizzi il boot manager LILO:

image=/vmlinuz

root=/dev/ram0

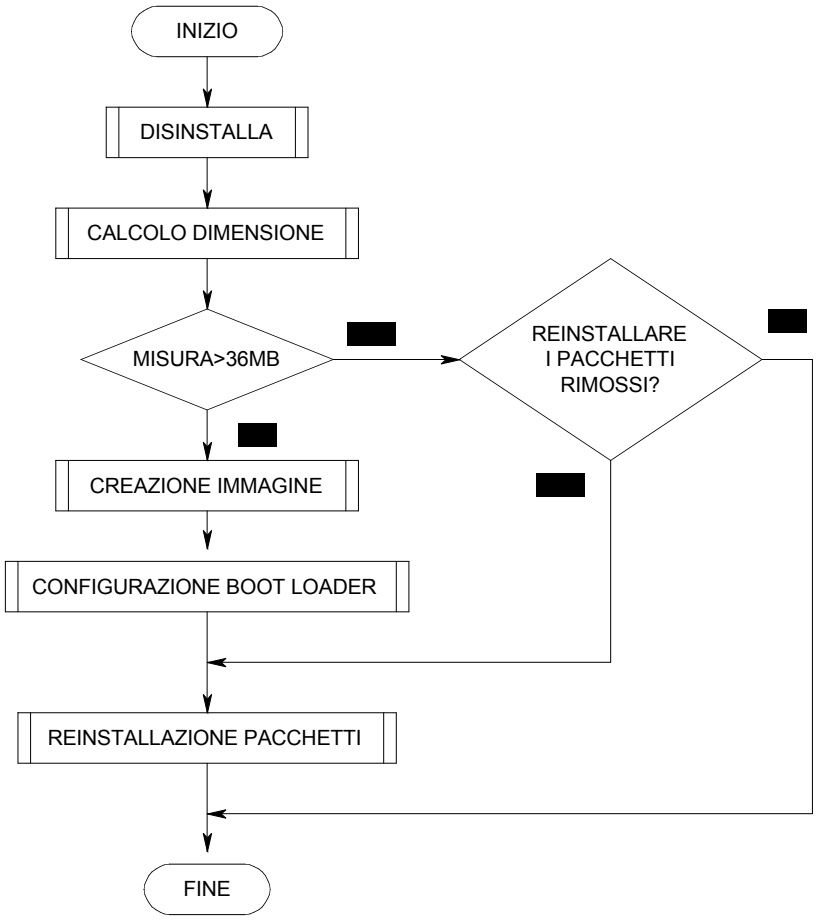
initrd=/…./image.gz

Per maggiori informazioni rimandiamo il lettore alla lettura del file initrd.txt contenuto nei sorgenti del kernel.

3.3.6. Creare l'immagine

Per creare una configurazione robusta si è preferito, in modalità operativa, caricare il file system root in un RAM disk (V:3.3.5,RAM disk come file system root). In modalità di sviluppo è possibile avviare lo script crea_immagine che provvede a creare l'immagine contenente il necessario per la modalità operativa basandosi sul file system della modalità di sviluppo. Il file immagine, ottenuto tramite una copia a basso livello, deve comprendere nella dimensione anche lo spazio libero. A questo scopo lo script accetta un argomento facoltativo in formato numerico che rappresenta la quantità, espressa in kilobyte, di spazio vuoto da riservare. Questo script si incarica inoltre di copiare il tutto sull'Hard Disk interno e di configurare il boot loader. Come è stato più volte specificato, la modalità operativa non necessita dello strato "strumenti per lo sviluppo", e questo comporta in modo particolare l'esclusione della directory /sviluppo e di alcuni pacchetti dalla creazione dell'immagine.

3.3.6.1 Diagramma di flusso di crea_immagine



Analisi dei blocchi funzionali

Disinstalla

Si effettua la disinstallazione dei pacchetti che non devono essere inclusi nella modalità operativa e che appartengono propriamente allo strato “strumenti per lo sviluppo”.

Calcolo dimensione

Volendo creare un file immagine senza spreco di spazio, si calcola la dimensione effettiva della parte di file system da copiare e tale valore viene posto nella variabile MISURA.

Creazione immagine

Si procede alla realizzazione vera e propria dell’immagine. Il risultato di questa operazione è un file compresso image.gz che contiene il file system root che verrà utilizzato in modalità operativa. I passi eseguiti sono i seguenti:

- copia del necessario in un file che viene chiamato image.tar: viene escluso il contenuto della directory /sviluppo e della directory /proc:
  tar -cvf /sviluppo/archivio/image.tar --exclude=/sviluppo/* --exclude=/proc/* /*
- espansione di image.tar in un RAM disk opportunamente dimensionato (la dimensione è contenuta nella variabile MISURA):
  mke2fs /dev/ram1 $MISURA
  mount /dev/ram1 /mnt
  cd /mnt
  tar -xvf /sviluppo/archivio/image.tar
- creazione dell’immagine del RAM disk attraverso l’utilizzo del comando “dd”:
  dd if=/dev/ram1 of=/sviluppo/archivio/image bs=1k count=$MISURA
- compressione di tale immagine nel file image.gz per ridurre lo spazio occupato:
  gzip /sviluppo/archivio/image

Configurazione boot loader

Una volta creata l’immagine bisogna fare in modo che il boot loader al riavvio successivo la utilizzi e per far questo va riconfigurato. Sull’Hard Disk interno vengono copiati alcuni file necessari al buon funzionamento di LILO:

- boot/vmlinuz: kernel utilizzato al caricamento della modalità operativa
- boot/kernelsviluppo : kernel utilizzato al caricamento della modalità di sviluppo
- boot/image.gz: immagine della root della modalità operativa
- boot/lilo.conf: file di configurazione di LILO
- dev/hard disk* : tutti i descrittori degli Hard Disk
- dev/ram* : tutti i descrittori dei RAM disk

Senza questi file non sarebbe possibile installare il boot loader LILO nell’MBR

Reinstallazione pacchetti

Poiché all'avvio dello script ci troviamo in modalità di sviluppo è necessario ripristinare la configurazione originale reinstallando i pacchetti che sono stati rimossi nel blocco funzionale "DISINSTALLA".

3.3.6.2 Listato commentato dello script crea_immagine

```
#!/bin/bash
#
# giugno 2003
# versione 3.0
# Questo script crea l'immagine della root necessaria alla modalità operativa.
umount -a
cd /sviluppo/script
# Si procede alla disinstallazione dei pacchetti non necessari alla modalità operativa.
# In questo modo non saranno inclusi nell'immagine.
# Il tutto viene eseguito dallo script "disinstalla"
/sviluppo/script/disinstalla
# Poiché l'immagine che si vuole creare verrà caricata in un RAM disk
# dobbiamo determinare la dimensione del RAM disk basandosi
# sull'effettivo spazio occupato dal file system
# MISURA contiene lo spazio occupato.
CONST=$(df)
CONTA=$[1]
for TEMPO in $CONST
do
if [ $CONTA = 10 ]
  then
  MISURA=$[$TEMPO]
fi
CONTA=$[$CONTA + 1]
done
if [ ! $1 ]
  then
  AGGIUNTI=$[0]
```

```
else
  AGGIUNTI=$1
fi
# Volendo escludere dall'immagine la directory /sviluppo
# qui si calcola la sua dimensione per non commettere errori nella
# stima dello spazio necessario
# SPAZIO contiene lo spazio occupato da /sviluppo
PACCHI=$(du -s /sviluppo)
CONTA=$[0]
for TEMPO in $PACCHI
do
if [ $CONTA = 0 ]
  then
  SPAZIO=$[$TEMPO]
fi
CONTA=$[1]
done
#  A MISURA viene aggiunto lo spazio AGGIUNTI indicato
#  come parametro all'avvio dello script
# Questo serve a sovradimensionare il RAM disk in modo
# da ottenere dello spazio libero pari ai kilobyte  indicati in AGGIUNTI
MISURA=$[$MISURA + $AGGIUNTI]
# A MISURA viene tolto SPAZIO cioè lo spazio occupato dalla directory /sviluppo
MISURA=$[$MISURA - $SPAZIO]
# Nel calcolo delle dimensioni del RAM disk dobbiamo
# aggiungere 1843 kb che normalmente sono riservati, dal sistema operativo,
# per il super user
MISURA=$[$MISURA + 1843]
# Controllo di dimensione massima del RAM disk,
# il sistema è configurato per avere al massimo un RAM disk di 36MB
if [ 36864 < $MISURA ]
  then
```

```
  echo "L'immagine risulterà maggiore del RAM disk che dovrà contenerla."
  echo "L'attuale misura dell'immagine è di $MISURA Kb"
  echo "È necessario diminuirne le dimensioni a meno di 36864 Kb"
  echo "Reinstallare i pacchetti rimossi (y/n)?"
  read AVANTI
  if [ ! $AVANTI = n ]
        then
        cd /sviluppo/script
        ./installa
  fi
exit
fi
# Raccolta dei file per l'immagine
# Grazie all'applicazione tar possiamo raccogliere
# solo la parte di file system che ci interessa.
tar -cvf /sviluppo/archivio/image.tar --exclude=/sviluppo/* --exclude=/proc/* /*
# Creazione di un RAM disk appropriato,
# utilizzato come buffer temporaneo per i dati da trasferire nell'immagine
mke2fs /dev/ram1 $MISURA
# I file precedentemente selezionati vengono decompressi in un RAM disk.
mount /dev/ram1 /mnt
cd /mnt
tar -xvf /sviluppo/archivio/image.tar
rm /sviluppo/archivio/image.tar
# Modifica della tabella delle partizioni
cp  /sviluppo/archivio/fstab_default   /mnt/etc/fstab
cd  /
# smontaggio del RAM disk
umount  /mnt
# Copia del contenuto del RAM disk nel file /sviluppo/archivio/image
dd if=/dev/ram1 of=/sviluppo/archivio/image bs=1k count=$MISURA
# Compressione dell'immagine appena creata
```

echo "Compressione file in corso; l'operazione potrebbe durare alcuni minuti"

gzip /sviluppo/archivio/image

#Copia dei file necessari:

 # LILO non permette alcuna installazione nell'MBR se non sono presenti specifici file

mount /dev/hda1 /mnt

cd /mnt

mkdir boot

mkdir dev

mkdir mnt

cd /

cp -Rf /dev/hd* /mnt/dev

cp -Rf /dev/ram* /mnt/dev

cp /sviluppo/archivio/vmlinuz /mnt/boot

cp /sviluppo/archivio/vmlinuz /mnt/boot/kernelsviluppo

cp /sviluppo/archivio/boot.b /mnt/boot

cp /sviluppo/archivio/lilo.conf /mnt/boot

cp /sviluppo/archivio/image.gz /mnt/boot

rm /sviluppo/archivio/image.gz

umount /mnt

 # Update del bootloader mediante l'avvio dello script lilo_update

/sviluppo/script/lilo_update

 # Reinstallazione dei pacchetti precedentemente disinstallati:

 # anche qui viene richiamato uno script

cd /sviluppo/script

./installa

 # FINE crea_immagine

3.3.7. Altri script

Vengono presentati i listati commentati di altri script. I più sono utilizzati dallo stesso crea_immagine. Tutti gli script qui menzionati sono contenuti nella directory /sviluppo/script nel file system root della modalità di sviluppo.

3.3.7.1 Script disinstalla

#!/bin/bash

```
# giugno 2003
#   versione 3.0
# Salvataggio librerie gcc per demone ssh: per evitare la loro cancellazione
# in una delle disinstallazioni successive
cp /lib/libgcc_s.so /lib/marmot.so
cp /lib/libgcc_s.so.1 /lib/marmot.so.1
# Con queste istruzioni otteniamo la rimozione dei pacchetti non utilizzati dalla
# modalità operativa
removepkg perl-5.6.1-i386-1
removepkg findutils-4.1.7-i386-1
removepkg ncurses-5.2-i386-1
removepkg kernel-headers-2.4.20-i386-5
removepkg glibc-2.3.1-i386-3
removepkg mawk-1.3.3-i386-1
removepkg gcc-3.1-i386-1
removepkg binutils-2.11.2-i386-1
removepkg make-3.79.1-i386-1
removepkg automake-1.6.2-i386-1
removepkg autoconf-2.53-i386-1
# Ripristino libreria gcc per demone ssh
mv /lib/marmot.so /lib/libgcc_s.so
mv /lib/marmot.so.1 /lib/libgcc_s.so.1
echo "FINE"
# Fine disinstalla
```

3.3.7.2 Script installa

```
#!/bin/bash
# giugno 2003
#   versione 3.0
# Con questo script otteniamo l'installazione dei pacchetti necessari
# alla compilazione, è importante non invertire l'ordine!!!!!!
# Questi pacchetti vano installati solo nella modalità di sviluppo
cd /sviluppo/packages
# pacchetti presi dalla distribuzione white dwarf 1.2.0
```

installpkg autoconf-2.53-i386-1.tgz

installpkg automake-1.6.2-i386-1.tgz

installpkg make-3.79.1-i386-1.tgz

installpkg binutils-2.11.2-i386-1.tgz

installpkg gcc-3.1-i386-1.tgz

installpkg mawk-1.3.3-i386-1.tgz

 # pacchetti presi dalla distribuzione Slackware 9.0

installpkg glibc-2.3.1-i386-3.tgz

installpkg kernel-headers-2.4.20-i386-5.tgz

 # pacchetti presi dalla distribuzione white dwarf 1.2.0

installpkg ncurses-5.2-i386-1.tgz

 # pacchetti presi dalla distribuzione Slackware 9.0

installpkg findutils-4.1.7-i386-1.tgz

installpkg ./perl-5.6.1/perl-5.6.1-i386-1.tgz

cd /sviluppo/script

 # Fine installa

3.3.7.3 Script lilo_update

#

 # giugno 2003

 # versione 3.0

 # Questo script aggiorna il boot manager LILO

 # Viene montata la partizione dell'Hard Disk interno e la seconda partizione

 # dell'Hard Disk esterno in modo da rispettare le direttive indicate nel file lilo.conf

mount /dev/hda1 /mnt

mount /dev/hdb2 /mnt/mnt

 # Si esegue l'aggiornamento del boot loader installandolo nell'MBR.

lilo -r /mnt -C /boot/lilo.conf -i /boot/boot.b

umount -a

 # fine lilo_update

Per completare il discorso introdotto con il listato di lilo_udate di seguito è stato inserito il listato commentato del file di configurazione lilo.conf:

 # giugno 2003

```
#          versione 3.0
#
#  Vengono definite le direttive globali
boot = /dev/hda # installazione nell'MBR
timeout=100          # tempo di attesa 10 secondi
vga = normal         # impostazione grafica
ramdisk=36864        # indica al kernel la dimensione massima dei
                     # RAM disk che deve supportare
default=Operativo    # fissa quale sia la voce da avviare
                     #dopo che è scaduto il tempo timeout
#  Permette all'utente di selezionare una delle voci del menu
prompt
# Configurazione di avvio della modalità operativa
image =/boot/vmlinuz
  root=/dev/ram0
 initrd=/boot/image.gz
  label=Operativo
  read-only
# Configurazione di avvio della modalità di sviluppo
image = /boot/kernelsviluppo
  root = /dev/hdb1
  label =Sviluppo
  read-only
# Configurazione di avvio della modalità di ripristino
image = /mnt/boot/vmlinuz
  root=/dev/hdib2
  label=Ripristino
  read-only
#fine lilo.conf
```

3.3.7.4 Script kernel_update

Deve essere utilizzato nella modalità di sviluppo e permette di aggiornarne il kernel con quello indicato come argomento dall'utente. L'operazione di aggiornamento è fatta in modo da influenzare solamente la modalità di sviluppo, questo permette all'utente, con un semplice riavvio, di verificare la funzionalità delle modifiche apportate. Lo

script prevede il backup del kernel prima della sua sostituzione, ma non prevede backup successivi, ciò significa che se utilizzato due volte con lo stesso kernel, viene sovrascritto sia il kernel attuale sia quello salvato.

```
#
#       giugno 2003
#       versione  3.0
#
#  Controllo dell'esistenza dell'argomento contenente il path
# e il nome del kernel da installare
if [ ! $1 ]
then
        echo "Uso: kernelupdate PATH"
        echo "Dove:"
        echo "PATH = percorso completo per il kernel da installare"
else
        # Backup del kernel
        cp /sviluppo/archivio/vmlinuz /sviluppo/archivio/vmlinuz.old
        # Viene fatta una copia del kernel indicato dall'utente
        cp $1 /sviluppo/archivio/vmlinuz
        # Viene sostituito il kernel con quello nuovo
        mount /dev/hda1 /mnt
        mount /dev/hdb2 /mnt/mnt
        cp /sviluppo/archivio/vmlinuz /mnt/boot/kernelsviluppo
        umount /mnt/mnt
        umount /mnt
        # Si aggiorna il boot manager utilizzando lo script lilo_update
        /packages/immagine/lilo_update
        echo "Nuovo kernel installato;"
        echo " il vecchio kernel è stato rinominato
/sviluppo/archivio/vmlinuz.old "
fi
# fine kernel_update
```

3.3.7.5 Script ripristino

L'aggiornamento del kernel con una versione errata può provocare l'inaccessibilità alla modalità di sviluppo. Per risolvere questo problema sono stati previsti strumenti specifici: è possibile ripristinare il kernel vmlinuz.old, salvato dallo script kernel_update al momento dell'aggiornamento, richiamando lo script /ripristino/ripristino dalla modalità ripristino.

```
#!/bin/bash
#
#       giugno 2003
#       versione  3.0
# Questo script ripristina il kernel della configurazione sviluppo
mount /dev/hda1 /mnt
mount /dev/hdb1 /mnt/mnt
cp /mnt/mnt/sviluppo/archivio/vmlinuz.old
/mnt/mnt/sviluppo/archivio/vmlinuz
cp /mnt/mnt/sviluppo/archivio/vmlinuz /mnt/boot/kernelsviluppo
lilo -r /mnt -C /mnt/mnt/sviluppo/archivio/ripristino.conf -i
/mnt/boot/boot.b
umount /mnt/mnt
umount /mnt
echo "Kernel vmlinuz.old ripristinato"
echo "Per completare la procedura:"
echo "1-Riavviare il sistema"
echo "2-Avviare la modalita' sviluppo (ripristino non sara' disponibile)"
echo "3-Eseguire lo script 'lilo_update' presente in /sviluppo/script"
echo "E' consigliabile annotare su carta i tre passi precedenti."
# fine ripristino
```

Come si può vedere dal listato lo script prevede la riconfigurazione del boot loader, di seguito si può esaminare il file ripristino.conf utilizzato come file di configurazione per LILO:

```
#       giugno 2003
#       versione 3.0
boot = /dev/hda
timeout = 100
vga = normal
```

```
ramdisk=36864
default=Operativo
prompt
# Configurazione di avvio della modalità operativa
image =/boot/vmlinuz
 root=/dev/ram0
 initrd=/boot/image.gz
 label=Operativo
 read-only
# Configurazione di avvio della modalità di sviluppo
image = /boot/kernelsviluppo
 root = /dev/hdb1
 label =Sviluppo
 read-only
# fine ripristino.conf
```

4. Modalità operative

4.1. Manuale per l'utente

4.1.1. Introduzione

All'avvio del sistema il boot loader garantisce l'accesso in tre modalità: operativa, di sviluppo e di ripristino.

La modalità operativa permette l'utilizzo del PC104, ma non permette di modificare la sua configurazione: ogni modifica sarà persa all'avvio successivo. Tale tipo di accesso è stato predisposto per essere utilizzato nella fase operativa del robot MARMOT, vale a dire quando non si è connessi ad alimentazioni e a supporti di memorizzazione esterni, in quanto improvvisi cali di tensione potrebbero danneggiare un normale file system.

La modalità di sviluppo, appoggiandosi all'Hard Disk esterno, ci permette di accedere ad un'installazione configurabile; eventuali modifiche sono consigliate solo ad utenti esperti o in caso di necessità. In tale fase si prevede che l'utente alimenti il PC104 attraverso la rete fissa (questo garantisce che non ci siano improvvise interruzioni di corrente), e utilizzi l'Hard Disk esterno per avere maggior spazio su cui lavorare. L'utente può apportare le modifiche che ritiene più appropriate, utilizzando le normali procedure del sistema operativo linux (installazione di pacchetti, compilazione del kernel….) fino ad ottenere il sistema desiderato; a questo punto, attraverso uno script, potrà sostituire la vecchia modalità operativa con quella appena configurata.

Nei prossimi paragrafi affronteremo in dettaglio i due tipi di accesso fin qui menzionati. Chi sia interessato al semplice utilizzo del sistema in modalità operativa può leggere semplicemente il paragrafo ad esso associato.

Infine la modalità di ripristino permette di entrare in un sistema sicuro in caso che le altre due modalità non si avviino, in modo da poter apportare cambiamenti alle configurazioni per riparare delle eventuali modifiche dannose.

4.1.2. Componenti necessari

Per avviare la modalità operativa è necessario solamente disporre del PC104 collegato ad una alimentazione in corrente continua a 5V.

Per le modalità di sviluppo e di ripristino occorre inoltre collegare un hard disk esterno e quindi fornire due alimentazioni in corrente continua, a 5 e 12 V.

In tutte le configurazioni è inoltre possibile collegare una scheda PCMCIA per il collegamento in rete wireless ed un lettore di floppy disk, ricordando che anche quest'ultimo necessita di alimentazione a 12 V.

4.1.3. Modalità d'accesso operativa

Componenti necessari:

PC104 collegato ad una alimentazione in corrente continua di 5V.

Componenti opzionali:

Tastiera

Monitor

Hard Disk esterno con alimentazioni a 5 e 12V

Floppy disk esterno con alimentazioni a 5 e 12V

Scheda PCMCIA per il collegamento wireless

Collegamento a rete ethernet

Utilizzo:

È possibile accedere in tale modalità selezionando la voce "OPERATIVO" quando, all'avvio, il boot loader LILO lo richiede; lo stesso risultato si ottiene attendendo alcuni secondi dopo la comparsa del menu di selezione senza aver premuto alcun tasto.

Viene avviato un normale sistema operativo linux derivato dalla distribuzione maggiore Slackware chiamato White Dwarf 1.2.

Il sistema avviato è configurato per l'utilizzo di:

Periferiche:

Scheda di rete integrata

PCMCIA bridge

PCMCIA wireless

Usb

Programmi:

Demone ssh (pacchetto tcpip.tgz)

Demone telnet (pacchetto openssh.tgz)

L'utente può apportare modifiche a questa configurazione, ma tutte queste modifiche verranno perse all'avvio successivo; se ci fosse tale esigenza bisogna accedere al PC104 attraverso la modalità di sviluppo selezionabile nel menù di avvio e descritta nel paragrafo successivo.

Le interfacce di rete sono configurate con i seguenti indirizzi IP:

Interfaccia eth0 via cavo: 192.0.2.7

Interfaccia eth1 wireless: 192.0.2.8

Il segnale che indica il corretto funzionamento della scheda PCMCIA è il ripetersi di due suoni acuti subito dopo l'introduzione della scheda; poiché questa periferica è l'ultima ad essere configurata in fase di avvio, questo segnale può essere utile per indicare la conclusione della fase di inizializzazione del calcolatore, in caso non si utilizzi un monitor.

4.1.4. Modalità d'accesso per lo sviluppo

Componenti necessari:

PC104

tastiera

monitor

Hard Disk esterno

Fonte di alimentazione sicura di 5 e 12V (ad esempio, un alimentatore per PC collegato all'alimentazione di rete).

Componenti opzionali:

Floppy disk esterno con alimentazioni a 5 e 12V

Scheda PCMCIA per il collegamento wireless

Collegamento a rete ethernet

Utilizzo:

In questo paragrafo verranno introdotti nuovi concetti relativi alle modalità di accesso, e vengono considerati noti quelli spiegati nel paragrafo precedente.

Questa modalità è stata implementata per andare incontro all'utente che abbia necessità di modificare la configurazione usata nella modalità operativa. Può, ad esempio, essere necessario installare un pacchetto particolare o aggiornarne uno già presente: come abbiamo già visto non è possibile fare ciò mentre il sistema è operativo.

Ricordiamo all'utente che per accedere in questa modalità e necessario che sia collegato l'Hard Disk esterno apposito, descritto nel capitolo componenti, e che l'alimentazione sia di rete e non a batteria. L'Hard Disk esterno è richiesto perché contiene la partizione su cui è installata una distribuzione White Dwarf 1.2, concettualmente uguale a quella della modalità operativa, ma adattata allo sviluppo, contenente cioè i pacchetti necessari alla compilazione di moduli, kernel, ecc.. L'alimentazione di rete serve a garantire che non ci siano blackout improvvisi che compromettano l'integrità del file system. È possibile accedere in tale modalità selezionando la voce "SVILUPPO" quando il menu di avvio lo richiede.

Per illustrare le procedure di configurazione divideremo questo argomento in due paragrafi principali: "Come apportare modifiche al file system" e "Come apportare modifiche al kernel".

Come apportare modifiche al file system

L'utente interessato ad aggiungere pacchetti o a modificare il contenuto di directory o file basta che modifichi il contenuto dell'attuale file system root: infatti, questa è la struttura sorgente in base alla quale viene creata l'immagine utilizzata per il file system root della modalità operativa.

Nelle modifiche del file system dobbiamo segnalare all'utente che:

- Non occorre cancellare i pacchetti di software necessari per compilare; la loro installazione e disinstallazione è gestita in modo automatico dagli script e quindi vengono disinstallati prima di creare l'immagine e reinstallati successivamente per garantire i servizi di compilazione.
- Il contenuto della directory /sviluppo è escluso in modo automatico dalla creazione dell'immagine.
- Il file system che verrà copiato sull'Hard Disk interno non può comunque eccedere i 36 MB, comprensivi dello spazio libero che viene riservato.

Come apportare modifiche al kernel

L'utente può procedere secondo la prassi linux per compilare il kernel.

Per procedere all'installazione deve avviare lo script /sviluppo/script/kernel_update indicando come argomento il percorso unitamente al nome del nuovo kernel. Questa procedura aggiorna l'attuale kernel con quello indicato; riavviando il sistema in modalità operativa si può testare la sua funzionalità. Se si riscontrano problemi con il kernel installato e la modalità di sviluppo non si riavvia consigliamo all'utente di accedere al sistema in modalità di ripristino (V:4.2.4,Modalità d'accesso per il ripristino).

ATTENZIONE: L'utilizzo dello script kernel_update può comportare la perdita dell'attuale kernel se utilizzato due volte e ciò comporta l'impossibilità di usufruire delle funzionalità fornite dalla modalità ripristino.

Effettuate tutte le modifiche necessarie, e dopo averne testato il funzionamento, è possibile creare l'immagine del sistema così configurato entrando nella directory /sviluppo/script e avviando lo script crea_immagine; questo script accetta un argomento opzionale in formato numerico che indica quanto spazio libero riservare,

espresso in kilobyte, nel file system che si sta creando, fermo restando che non è possibile superare i 36MB complessivi.

In conclusione vorremmo che fosse chiaro che per preparare il sistema ottimizzato e desiderato da utilizzare in modalità operativa, è prima di tutto necessario che funzioni allo stesso modo e nella stessa configurazione la modalità di sviluppo.

4.1.5. Modalità d'accesso per il ripristino

Componenti necessari:

PC104

tastiera

monitor

Hard Disk esterno

Fonte di alimentazione sicura di 5 e 12V (ad esempio, un alimentatore per PC collegato all'alimentazione di rete).

Componenti opzionali:

Floppy disk esterno con alimentazioni a 5 e 12V

La modalità d'accesso per il ripristino permette all'utente di porre rimedio ad un'errata configurazione della modalità di sviluppo quando questa non si avvia. In particolare ci si può trovare in questa situazione se il kernel compilato e sostituito presenta grosse incompatibilità con l'intero sistema. Dopo aver avviato il sistema in modalità ripristino, basta avviare lo script chiamato "ripristino" presente nella directory "/ripristino", che provvede a reinstallare il kernel precedente e ad aggiornare il boot loader.

4.2. Configurazione delle periferiche

Rispetto all'installazione originale della distribuzione White Dwarf è stato necessario impostare molti parametri di avvio per accordarli alle nostre esigenze; in particolare si sono dovute configurare le interfacce di rete per adattarle a quelle del laboratorio e ricompilare un nuovo kernel.

Buona parte del lavoro di configurazione riguarda proprio la compilazione del kernel; discutere di tutte le scelte di compilazione sarebbe un lavoro molto lungo ed anche inutile, quindi diamo una spiegazione solo dei parametri più importanti: nella documentazione viene accluso anche il file di configurazione dove sono specificate tutte le scelte effettuate. Abbiamo scelto di compilare un kernel monolitico, nel senso che abbiamo incluso nel kernel tutte le opzioni per il controllo diretto delle periferiche senza il bisogno di caricare moduli aggiuntivi, per limitare lo spostamento di file tra un hard disk e l'altro.

Ecco una breve spiegazione delle opzioni scelte per le parti principali del kernel:

Code maturity level options: viene impostato per mostrare anche le opzioni di tipo sperimentale.

Loadable module support: viene abilitato il supporto di gestione per eventuali moduli esterni al kernel, come quelli che saranno necessari per la scheda PCMCIA.

Processor type and features: il kernel viene compilato per un Pentium MMX con coprocessore matematico e viene disabilitato il supporto per sistemi multiprocessore.

General setup: viene incluso il supporto per i bus PCI, ISA, EISA, e il supporto HOTPLUG per il bus USB. Viene inoltre specificato che verranno attivate le funzionalità di rete.

PCMCIA/Cardbus support: vengono abilitate le gestioni di schede PCMCIA e Cardbus, insieme alla gestione di bridge compatibili i82365.

Parallel port support: viene abilitata la gestione della porta parallela.

Plug and Play configuration: viene abilitata la gestione automatica delle periferiche plug and play.

Block devices: viene abilitato il driver per floppy disk, il supporto per un RAM disk iniziale di 32768 KB e il riconoscimento in avvio della funzione initrd

Networking Options: vengono abilitati i servizi di rete più comuni.

IDE, ATA and ATAPI block devices: viene abilitato il supporto per i dispositivi IDE come floppy, hard disk, Cdrom e dispositivi con interfaccia IDE/SCSI; viene inoltre attivato il controllo del chipset ALI presente sulla scheda madre del PC104. Bisogna porre particolare attenzione all'opzione "Use old IDE driver on primary interface only": questa opzione che abbiamo selezionato carica un vecchio driver IDE per il controllo del primo disco fisso, la memoria flash interna, che non sembra funzionare bene con i nuovi driver. Questa opzione rende però il kernel inutilizzabile su sistemi che usano come primo hard disk un dispositivo normale, e purtroppo non è sicuro se le future versioni dei kernel linux continueranno a supportare questo driver.

SCSI Support: il supporto SCSI generico è stato abilitato per poter gestire i dispositivi di memoria di massa collegabili tramite USB, che vengono appunto gestiti come dispositivi SCSI

Network devices support: sono stati abilitati i supporti per i driver e100 della scheda di rete e il supporto generico per schede PCMCIA di trasmissione wireless.

Character devices: sono stati abilitati i controller delle porte seriali di trasmissione.

Multimedia devices: è stato abilitato il supporto a "Video for Linux", necessario per attivare poi i driver della webcam.

File systems: i tipi di file system che sono stati abilitati sono: file system MAC, ext2, ext3, DOS, UMSDOS, FAT, ISO9660 e Joliet in modo completo e NTFS in sola lettura. In questo modo dovrebbe essere possibile scaricare dati da quasi ogni fonte sul PC104.

Partition types: oltre al partizionamento effettuato con sistemi linux, vengono riconosciuti dischi che sono stati partizionati sotto DOS.

USB Support: il controller della HUB USB del PC104 non è standard e richiede quindi il driver OHCI per funzionare. Sono inoltre stati abilitati il supporto per memorie di massa come le flashpen e i driver per le webcam Philips come richiesto del gruppo che si occupa della webcam.

Oltre a ciò si sono dovuti sovrascrivere alcuni dei file sorgenti del kernel prima di compilarlo; il gruppo MARMOTCAM ci ha segnalato la necessità che il driver PWC per la webcam Philips fosse aggiornato almeno alla versione 8.8, mentre la versione distribuita con i sorgenti del kernel 2.4.20 è solamente una 8.6. Abbiamo quindi scompattato il file pwc-8.8.tar.gz in una directory e sovrascritto con i file così ottenuti la sottodirectory /drivers/usb dei sorgenti del kernel.

Per configurare la tastiera si è provveduto a copiare la mappa dei caratteri di una tastiera italiana con il comando "loadkeys –m it > /mnt/kernel-2.4.20/linux-2.4.20/drivers/char/defkeymap.c" (questo percorso è valido se la partizione dei sorgenti è montata nella directory /mnt) lanciato prima del comando "make dep".

Un grosso problema è stato l'installazione dei driver per la scheda PCMCIA; i sorgenti scaricati dal web sono infatti configurati per cercare ad ogni costo di utilizzare tutti i servizi già compilati nel kernel senza compilare altri moduli, anche se questi sarebbero invece necessari.

Il problema è stato aggirato compilando i sorgenti secondo la normale procedura e integrando con la compilazione diretta dei moduli: è necessario entrare nella sottodirectory chiamata wireless e lanciare il comando "make all", che provvede a compilare tutti i moduli qui presenti. Ottenuti i moduli questi sono stati copiati nella directory /lib/modules/2.4.20/pcmcia, dove i programmi di gestione dei driver possono trovarli correttamente.

Un'altra richiesta del gruppo MARMOTCAM è stata l'installazione del demone ssh, che siamo stati costretti a compilare dato che non è incluso nella distribuzione White Dwarf. Per compilarlo, dopo aver scompattato il file openssh-3.4p1.tar.gz è necessario copiare lo script wd.config_ssh nella directory che viene creata, spostarsi in quest'ultima e rendere tale script eseguibile tramite il comando "chmod +x wd.config_ssh". Dopo aver eseguito tale script sarà sufficiente digitare "/usr/local/sbin/sshard disk –b 1024" per mandare il demone in esecuzione.

Installato il nostro kernel e i driver sono stati poi modificati i file di configurazione del sistema, al fine di ottenere i seguenti risultati:

4.2.1. Impostazione bridge ISA to PCMCIA

File: "/etc/rc.d/rc.pcmcia"

Modifica riga 38

PCIC=i82365

Questa riga istruisce il sistema riguardo al bridge ISAtoPCMCIA affermando che esso è compatibile con l'interfaccia Intel 82365.

4.2.2. Impostazione PCMCIA wireless

Impostazione degli indirizzi IP scheda PCMCIA wireless

File: “/etc/network.opts”

Modifica righe 26-31:

IPADDR=”192.0.2.8”

NETMASK=”255.255.255.0”

NETWORK=”192.0.2.0”

BROADCAST=”192.0.2.255”

 # Gateway address for static routing

GATEWAY=”192.0.2.1”

Queste modifiche impostano gli indirizzi IP e l’instradamento della scheda wireless; il loro significato è del tutto analogo a quello delle modifiche del file rc.inet1. L’indirizzo IP della scheda PCMCIA è stato settato su 192.0.2.8

Impostazione del modulo orinoco_cs

File: “/etc/modules.conf”

Modifica riga 13:

alias eth1 orinoco_cs

Viene specificato che l‘interfaccia di rete eth1 corrisponde al dispositivo supportato dal modulo “orinoco_cs”, quello che viene caricato per l’utilizzo della scheda PCMCIA.

Impostazione dell’instradamento

File: ”/etc/network”

Modifica riga 60:

ifconfig eth0 down

Modifica righe 269-270:

ifconfig eth0 up

route add default gw $GATEWAY

Queste due aggiunte al file servono a risolvere un problema di instradamento di pacchetti sulla rete; infatti, ogni volta che la scheda PCMCIA viene inserita essa diventa l’interfaccia di default verso la rete.

Se così non fosse verrebbe invece utilizzata di preferenza eth0, che viene tenuta attiva anche quando il cavo di rete non è collegato, e i pacchetti non verrebbero mai trasmessi qualora il cavo fosse rimosso ma fosse presente la scheda wireless.

Impostazione degli IRQ

File:“/etc/config.opts”

Modifica righe 22-23 :

 # Second built in serial port

exclude irq 3

Viene riservata la linea di interrupt numero 3 per la seconda porta seriale del PC 104, che altrimenti potrebbe essere utilizzata dalla scheda PCMCIA generando un conflitto.

4.2.3. Impostazione scheda ethernet integrata

Impostazione degli indirizzi IP scheda di rete integrata

File: ”/etc/rc.d/rc.inet1”

Modifica righe 24-29:

IPADDR=”192.0.2.7”

NETMASK=”255.255.255.0”

NETWORK=”192.0.2.0”

BROADCAST=”192.0.2.255”

 # Gateway address for static routing

GATEWAY=”192.0.2.1”

Queste modifiche impostano gli indirizzi IP e l’instradamento della scheda ethernet secondo i dati della rete del laboratorio. L’indirizzo IP della scheda ethernet da noi settato è 192.0.2.7

4.2.4. Script eseguiti all’avvio

File: ”/etc/rc.d/rc.3”

Modifica righe 7-9:

/etc/rc.d/rc.hotplug start

/etc/rc.d/rc.pcmcia start

/etc/rc.d/rc.marmot

Questo file è uno script completamente creato dal nostro gruppo, che viene eseguito durante la fase di avvio del sistema. La prima riga avvia il supporto per il rilevamento automatico della connessione di periferiche sul bus USB, la seconda riga avvia il programma per la gestione della scheda PCMCIA mentre la terza riga richiama l’ulteriore script creato da noi rc.marmot.

File: ”/etc/rc.d/rc.marmot”

Tutto il file è stato modificato; questo è l’attuale contenuto:

```
#!/bin/bash
#       Questo script viene richiamato all'avvio da rc.3
echo "E' in esecuzione /etc/rc.d/rc.marmot.....attendere "
mount /dev/hda1 /flash
if [ -x "/flash/initmarmot" ]
                        then
```

```
        /flash/initmarmot
    else
        echo "IMPOSSIBILE ESEGUIRE /flash/initmarmot"
fi
route add default gw 192.0.2.1
/usr/local/sbin/sshd -b 1024
```

Questo script viene eseguito appena prima della chiusura della procedura di avvio; esso monta il contenuto dell'Hard Disk interno nella directory /flash e cerca di mandare in esecuzione lo script qui contenuto e denominato "initmarmot".

Le ultime due righe servono per configurare il gateway della rete e per mandare in esecuzione il demone ssh.

4.3. Avvertenze

Pur essendo un dispositivo a basso consumo il PC104 sembra essere soggetto ad un surriscaldamento piuttosto veloce.

Abbiamo notato che dopo un paio d'ore di uso continuato scattano gli allarmi di calore interni e il sistema comincia a riavviarsi improvvisamente ogni pochi minuti, finché non viene lasciato spento per una decina di minuti.

Questo problema sembra essere amplificato dall'uso della scheda PCMCIA che sembra avere un consumo, e il relativo riscaldamento, relativamente alti.

Secondo il nostro gruppo sarebbe consigliabile trovare un metodo per raffreddare il processore e la scheda PCMCIA, ad esempio praticando una finestra nella scatola che protegge il PC104 e che ora è priva di aperture, e montando su di essa una ventola per la circolazione dell'aria.

5. Conclusioni e sviluppi futuri

Al termine di questo lavoro il PC104 funzionava regolarmente e tutti i servizi richiesti erano garantiti. Sono possibili ulteriori miglioramenti soprattutto in termini di spazio occupato dal sistema e di miglioramenti agli script di gestione da noi realizzati. In particolare il demone ssh serve solo per criptare le comunicazioni mentre occupa molto spazio; sarebbe interessante cercare di lanciare comandi in remoto tramite telnet.

Siamo inoltre venuti a conoscenza che pochi giorni dopo la fine di questo elaborato è stata rilasciata la versione 2.0 della distribuzione White Dwarf, scaricabile dal sito www.whitedwarflinux.com.

Questa nuova versione dovrebbe comprendere un demone ssh più piccolo di quello da noi compilato e inoltre dovrebbe contenere dei pacchetti più aggiornati.

6. APPENDICI

6.1. APPENDICE A: Descrizione dettagliata dei componenti

6.1.1. A1: il PC104

Il PC104 da noi utilizzato è composto di un sistema MOPSlcd-6 della JUMPtec Industrielle Computertechnik equipaggiato con un chipDISK-IDE da 32 MB prodotto dalla stessa industria e con 64 MB di memoria RAM.

Ecco i suoi dati tecnici:

Processore: Intel Pentium 166 MMX

Ram: 1 SIMM da 64 MB

ChipDISK-IDE: ChipDISK/32-IDE della JUMPtec Industrielle Computertechnik da 32 MB

Interfaccia di rete Ethernet 10/100 BaseT

Interfaccia per disco floppy

Interfaccia EIDE con modalità ultraDMA 33

Interfaccia parallela con supporto per gli interrupt

2 porte seriali RS232

1 porta USB 1.1

1 porta per tastiere standard

Scheda grafica PCI della C&T con 2Mb di RAM video

Possibilità di connettere uno schermo LCD sull'apposito connettore interno

Timer Watchdog

Dimensioni 96x96 mm

Consumi tipici: 1,8 A a 5V

Temperatura operativa: da 0 a 60°C

Temperatura di stoccaggio: da -10 a 85°C

Umidità operativa: da 10% a 90%

Umidità di stoccaggio: da 5% al 95%

6.1.2. A2: la scheda PCMCIA

Dati tecnici:

Range di frequenza: 2400-2483.5 MHz

Canali di trasmissione: 4/11/13/14 (Può variare da nazione a nazione)

Modulazione: Direct Sequence Spread Spectrum (CCK, DQPSK, DBPSK)

Protocolli: CSMA/CA con ACK

Interfacce: PC card con adattatore ISA o PCI

Misure: 117.8x53.95x8.7 mm (PC Card)

Potenza segnale in uscita: 15 dBm

Alimentazione (+5V) : scheda inattiva 9 mA, scheda in ricezione 185 mA, scheda in trasmissione 285 mA, se usata con adattatore ISA -25mA.

Compatibilità: Novell Client 3.x & 4.x, Windows 95/98/NT/2000, Apple, Windows CE

Supporto per IEEE 802.11b HR standard

6.1.3. A3: Hard Disk esterno

Dati tecnici:

Fabbricato da: Caviar

Modello: AC22500

Capacità: 2559,8 Megabyte

Numero seriale: WT3490242464

Cilindri: 4960

Testine: 16

Settori: 63

Consumi a 5V: 0,48 A

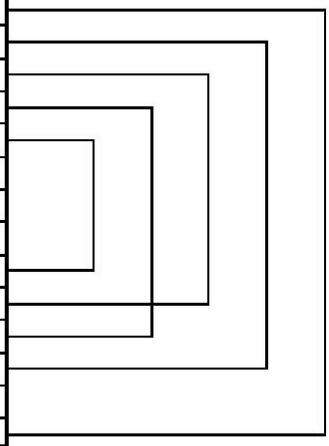
Consumi a 12V: 0,27 A

6.2. APPENDICE B: Aggiornamento del BIOS

Durante la realizzazione di questo progetto è stato necessario aggiornare il BIOS per ripristinare il funzionamento corretto del sistema; in questo paragrafo viene data una breve spiegazione della procedura seguita. Innanzi tutto è necessario scaricare dal sito della JUMPtec l'ultima versione del BIOS disponibile; essa viene fornita assieme ad altri file eseguibili in ambiente Windows e compressi in un file chiamato CRD2P588.ZIP. Una volta scompattato questo file in una cartella bisogna eseguire il file WinCris.exe che, attraverso una procedura guidata, porta alla creazione di un floppy disk con i seguenti file: Minidos.sys, Phlash.exe, Platform.bin e Bios.rom. A questo punto bisogna forzare il BIOS della scheda a leggere in avvio questo dischetto anche se la procedura di avvio non funziona più; per far ciò è necessario inserire nella porta parallela una chiave di update che può essere costruita collegando i segnali dell'interfaccia parallela secondo il seguente schema:

Function	Kontron 26 Pin No. *	D-Type-25 Pin No.
NSTROBE	1	1
Data0	3	2
Data1	5	3
Data2	7	4
Data3	9	5
Data4	11	6
Data5	13	7
Data6	15	8
Data7	17	9
nAck	19	10
Busy	21	11
Paper-Out / Paper-End	23	12
Select	25	13
nAuto-Linefeed	2	14
nError / nFault	4	15
nInitialize	6	16
nSelect-Printer / nSelect-In	8	17
Ground	10,12,14, 16, 18, 20, 22, 24	18-25

PLEASE CONNECT THE PINS AS FOLLOWS



A questo punto basta collegare al PC104 il lettore di floppy disk, inserire il dischetto creato precedentemente, infilare la chiave di update e avviare il sistema, avendo cura di togliere la chiave quando si accende il led di lettura del floppy disk.

Il sistema provvederà a tutta la procedura senza emettere segnali a video ma emettendo ogni tanto un suono acuto; al termine il sistema si riavvierà con il nuovo bios.

6.3. Appendice C: Note per la sostituzione del kernel

Il kernel è una tra le parti più importanti del sistema operativo e la sua sostituzione spesso porta a dover apportare cambiamenti anche al resto del sistema. Il PC104 non dispone di un processore molto veloce, quindi la compilazione di un kernel in versione 2.4.20 richiede circa un'ora di tempo. A questo tempo bisogna aggiungere quello necessario per ricompilare i moduli dei driver per la scheda PCMCIA; infatti i punti di innesto di questi file sono configurati per un solo kernel specificato alla compilazione e quindi essi vanno ricompilati ad ogni sostituzione del kernel. Ricordiamo inoltre che quando il boot loader LILO crea la propria mappa dei file da caricare in avvio (come i kernel), esso memorizza non la posizione che essi hanno nel file system bensì la posizione fisica sul disco (cilindro, testina, settore); perciò sovrascrivere tali file non è sufficiente a garantire l'avvio del sistema. Dopo la sostituzione dei file di avvio è quindi sempre necessario aggiornare il boot loader LILO (vedi LILO-HOWTO).

6.4. APPENDICE D: Creare un dischetto di avvio con il boot loader GRUB

Uno strumento che può essere utile per accedere a sistemi privi di boot loader è il dischetto di avvio con il boot loader GRUB; di seguito viene spiegato come crearne uno.

Per prima cosa occorre spostarsi nella directory in cui è stato copiato il pacchetto GRUB; a questo punto, usufruendo di un dischetto formattato, si procede coi seguenti comandi:

dd if=stage1 of=/dev/fd0 bs=512 count=1

dd if=stage2 of=/dev/fd0 bs=512 seek=1

Su questo dischetto non è possibile aggiungere altri file perché non è possibile montarlo.

Bibliografia

La documentazione che abbiamo utilizzato e che elenchiamo di seguito è scaricabile dal sito http://www.linuxdoc.org/

[1] Tom Fawcett: "The Linux Bootdisk HOWTO";

[2] Miroslav Skoric: "LILO mini-HOWTO";

[3] Greg O'Keefe: "From Power-up to Bash Prompt HOWTO";

[4] Robert Nemkin, Al Dev, Markus Gutschke, Ken Yap, Gero Kuhlmann: "Diskless Nodes HOWTO";

[5] Werner Almesberger, Hans Lermen: "Using the initial RAM disk (initrd)" ;

[6] Paul Gortmaker, "Using the RAM disk block device with Linux" ;

[7] Daniele Giacobini: "Appunti di informatica libera", Gennaio 2003;

È stata inoltre consultata la seguente relazione:

[8] Roberto Franchina, Vittorio Gregorelli: “Configurazione robusta del sistema operativo Linux con Ramdisk ed attivazione delle interfacce di rete”, elaborato d’esame di robotica, Gennaio 2002

Indice