

Getting Linux Help

How to use Linux man pages

Type "man command" where command is the name of the command you need help with. Once you are in the man page:

- Type "Q" anytime to quit.
- Type <space> to see the next page.

You can use the keyword function in man. To find commands that deal with dns, type "man -k dns". A listing of all commands system calls and other items that have the word "dns" in their name or short description is given.

man -k dns | grep domain Search man pages for occurrences of "dns" and "domain".

man 5 crontab See the man page in section 5 on crontab rather than the page in section 1.

To print a man page type "man name | lpr -P".

How to use info pages

There is a tutorial in the info pages that you can use to learn the commands. I recommend that before using info pages, you do at least a quick tour by taking the tutorial. Take notes and keep them handy when using the info pages. You may use the info pages by typing "info command" where command is the name of the command you need information about. To begin the tutorial, type "info", and look for the help section. Some of the basic info commands are listed below.

h	Take the help tutorial
<SPC>	To move down (see more text) on a screen
 or Backspace	To page up the screen
b	To move to the top of a screen
n	To move to the next node (text on a particular topic)
p	To move to the previous node
d	To move to the main directory node
l	To move to the last node you were at. Retraces where you were.
<CTRL>L	Refresh the screen
m	Show a menu of nodes you can move to
<CTRL>g	Cancel menu selection
name	Type node name or Ctrl-g to cancel after typing "m" for menu
u	To move back to the menu you were at. Use this command after making a menu selection to get back to the last menu from which you made a selection.
?	Show available commands.

How To use Linux Filesystems and Files

*How to make a symbolic link

The command `ln -s` creates a symbolic link to a file. For example, if you use the command

```
ln -s myfile pointer
```

you will create a symbolic link named "pointer" that points to the file "myfile". If you use "`ls -i`", the two files are listed with different inodes.

```
ls -i myfile pointer
180506 myfile 180507 pointer
```

By using "`ls -l`", the file "pointer" is shown as a symlink pointing to "myfile".

```
ls -l myfile pointer
lrwxrwxrwx 1 root root 5 Feb 28 17:18 here -> there
-rw-r--r-- 1 root root 5 Feb 28 17:17 there
```

+How to mount a DOS filesystem

1. Determine where your dos file system is (This assumes, you can boot to it using lilo).
 1. Type "`less /etc/lilo.conf`"
 2. Look for the label used by lilo to select dos, such as "dos" and use the associated device. The line on my system is "`other=/dev/hda1`" so I will use device "`/dev/hda1`".
2. Make a subdirectory "`/dos`" by typing "`mkdir dos`"
3. Do the mount by typing "`mount /mnt/hda1 /mnt/dos -t msdos`". To let all users have permission to use the dos partition, type "`mount -t vfat user,rw,exec,umask=000 /dev/hda1 /mnt/dos`".
4. Edit your `/etc/fstab` file to include:

```
/dev/hda1 /dos vfat defaults 0 0
```

How to determine file systems already mounted

Type "`less /etc/fstab`" or "`less /etc/mstab`". The contents of this file tells all filesystems, their type, etc. The `fstab` file lists files systems that are mounted when the system started. The `mtab` file is where the mount command stores the list of filesystems mounted.

+How to determine room used and left on filesystems

```
df - Disk free space.
du - Disk usage.
```

*How to set up and mount an NFS filesystem

Server Setup

To set up the server side, edit the file "`/etc/exports`" as in one of the examples below then type "`exportfs -a`". Also activate NFS services using `linuxconf`.

This is the first attempt which is an example of one way to do it:

```
/tftpboot/10.1.3.116          linux1(rw,no_root_squash)
/tftpboot/filesystems/usr *.mycompany.com(ro,root_squash)
```

```
/tftpboot                linux2(rw,no_root_squash)
/tftpboot                linux3(rw,no_root_squash)
/data                    linux4(rw,no_root_squash)
```

This is the way remote booting was set up for nfs:

```
/tftpboot/lts/ltsroot    10.1.0.101/255.255.0.0(ro,no_root_squash)
/tftpboot/lts/ltsroot    10.1.200.1/255.255.0.0(ro,no_root_squash)
/tftpboot/lts/ltsroot    10.1.200.2/255.255.0.0(ro,no_root_squash)
/tftpboot/lts/linux3     10.1.200.2/255/255.0.0(rw,no_root_squash)
```

For a remote boot machine, "linux3", after making a /tmp/mnt directory, type "mnt -n 10.1.0.100:/tftpboot/lts/linux3 /tmp/mnt -t nfs". The -n is only needed if the /etc directory is read only.

For a full explanation of the above options and when to use the no_root_squash option, read "The CTDLP Linux User's Guide".

Client Setup

To set up the client side on a fully functional Linux machine type "mount -o rsize=1024,wsize=1024 mymachine:/data /mnt/mymachine/data"

*How to format a linux floppy

1. Type "fdformat /dev/fd0H1440" to format the floppy.
2. To make a filesystem on the disk type "mkfs -t filesystem -c /dev/fd0H1440" where filesystem is the type of filesystem, usually ext2 (linux native).
3. Mount the filesystem "mount -t ext2 /dev/fd0 /mnt/floppy".
4. Use the filesystem by copying to/from /mnt/floppy.

*How to make a file system

Making a swap partition

```
type "mkswap -c /dev/hda3 10336"
```

The -c has swap check for bad blocks. The 10336 is the size of the partition in blocks, about 10M. The system enables swap partitions at boot time, but if installing a new system you can type "swapon /dev/hda3" to enable it immediately.

Making an ext2 file system on a floppy

1. fdformat /dev/fd0H1440
2. mkfs -t ext2 -c /dev/fd0H1440

Other file systems:

A normal hard drive can have many types of filesystems on it. To create an ext2 file system, type "mke2fs -c /dev/hda2 82080" to create an 82 meg filesystem. Note: mkfs is a front end to many file system types including ext2, minix, and msdos.

*How to check a file system

fsck - Used to check and repair a filesystem.

fsck is a front end to a filesystem type specific fsck.ext2, fsck.minix, and fsck.msdos.

Syntax: fsck -t type device

Ex: `fsck -t ext2 /dev/hda3`

The program `fsck` should never be run on a mounted filesystem with write permission since it can damage the filesystem. Unmount the filesystem before running `fsck` or be sure it is in read only mode.

***How to copy/remove a folder and all its contents with subdirectories**

To copy type "`cp -dpr sourcefile destinationfile`"

To remove type "`rm -rf directoryname`"

***Copying a drive**

```
dd if=/dev/hdb1 of=/backup/  
cp -dpr / /backup
```

***Finding files**

The `locate` command allows you to find any filename containing a string you type in. This is database driven, so it's fast. To initially load the database, do the following command, which will take a couple minutes or so to run:

```
slocate -u
```

Thereafter, type "`locate filename`" where "filename" is the name of the file you want to find.

The `whereis` command is very useful for finding binary programs and their associated man pages.

Redirecting Program output

The shell assigns the number 1 to standard output, and 2 to standard error output.

<code>gcc test.c >& errors</code>	Redirects stdout and stderr to the file errors.
<code>gcc test.c &> errors</code>	Same as above
<code>gcc test.c 2> errors</code>	Redirects error messages to the file errors
<code>gcc test.c 2>errors >/dev/null</code>	Redirects errors to the file errors and throws away the standard output messages.

See the `bash(1)` man page for more information on redirection of standard output and standard error.

Making a simple Emergency boot floppy

How to make a single boot floppy:

1. Find the kernel. It is usually `/vmlinuz` or `/boot/vmlinuz`. The file `vmlinuz` may be a softlink to the actual kernel executable. Find the executable kernel.
2. Copy the kernel image to the floppy.

```
dd if=/vmlinuz of=/dev/fd0
```

3. Type the command "`df`" and examine it's output to determine where your root filesystem is. Your root is "/" and is mounted on something like `"/dev/hda2"`.
4. Set the kernel image on the floppy to the location of your root system.

`rdev /dev/fd0 /dev/hda2`

Your root filesystem may be somewhere other than `"/dev/hda2"`.

5. Test the floppy by rebooting your system and attempting to boot from it. You can use the command `"badblocks /dev/fd01440 1440"` to check the floppy for badblocks.

Setting up Removable and External Filesystems to Automatically mount when used

1. The program "autofs" must be setup to run as a daemon upon system startup. To do this with Redhat Linux, use the program "linuxconf" and select "Control", "Control panel", "Control Service activity". Activate "autofs" using the menu selections.
2. Edit the file `"/etc/auto.master"` to the following:
 3. `/mnt /etc/auto.misc --timeout 20`

The above example sets the program to unmount the device after 20 seconds.

4. Edit the file `"/etc/auto.misc"` adding lines like:
 5. `cd -fstype=iso9660,ro :/dev/cdrom`
 6. `fl -fstype=auto :/dev/fd0`

This will cause the cd-rom to be mounted when you access the directory `"/mnt/cd"` and the floppy to be mounted when you access `"/mnt/fl"`. The directories `"/mnt/cd"` and `"/mnt/fl"` must not exist in order for this to work.

7. To use automount, put a cd in the CD-ROM drive and type `"ls /mnt/cd"` or `"cd /mnt/cd"`.

Runlevels

How to set the startup runlevel

1. Edit the file `/etc/inittab`
2. In the file is a line like:

`id:3:initdefault:`

In this case the value 3 specifies the runlevel. Normally you will want this to be 3 to run in console mode or 5 to run an X session at startup. A value of 1 is single user mode and can be set when having problems with your system. Modify this value to the value you want.

Below is a list of runlevels and what they mean.

- 0 - halt
- 1 - Single user mode
- 2 - Multiuser, without NFS (The same as 3, if you don't have networking)
- 3 - Full multiuser mode
- 4 - unused
- 5 - X Session
- 6 - Reboot

How to change runlevels while the system is running

A running system can be taken to single user mode by using the telinit command to request run level 1 as follows:

```
telinit 1
```

Any valid runlevel may be entered by typing this command followed by the desired runlevel value. Normally you will want to use 1, 3, or 5.

How To Manage Linux Users

Howto add a user

While logged in as root, type:

```
adduser username
```

Where "username" is the name of the user you want to add. Enter the password for the user when prompted.

How to change a password

While logged in as root or the user that will be changed, type:

```
passwd username
```

Where "username" is the name of the user whose password you want to change. If "passwd" is typed, the password will be changed for the user, you are logged in as.

How to remove a user

While logged in as root, type:

```
userdel -r username
```

Where "username" is the name of the user you want to remove. This will remove the user's home directory. You can delete the user without the "-r" option and delete the user's home directory manually. If the group the user was in, is no longer needed, you may delete it by editing the "/etc/group" file.

Determining Linux Dependencies

*How to compile the Linux kernel

1. Backup the present kernel which is in `/boot` for most systems. You can tell where it is by looking at `/etc/lilo.conf` . sometimes lilo.conf will point to a link file. Trace the link file and backup the file being pointed to.
2. Have an alternate way to boot such as another linux system on your computer or an emergency boot disk that you can access your filesystem from in case your new kernel crashes.
3. Back up your kernel modules in the directory `/modules` . There is probably a directory full of modules so you should be able to back it up using `cp -dpr dir1 dir2` where dir1 is the name of the directory where your modules are and dir2 is where you want to put them.
4. Go to `/usr/src/linux` . There is a `/usr/src/linux/.config` file used to do the compile. It is modified by typing `make config` and the values in it are used to determine defaults while doing the `make config` . Back this file up to keep your original settings if you want.
5. Type `make config` and answer the hundreds of questions asked. On another terminal, open the file `/usr/src/linux/Documentation/Configure.help` to determine what each configuration is. Also you can type `make menuconfig` or from an X session `make xconfig` .
6. Type `make dep`
7. Type `make clean`
8. Type `make bzImage` or `make zImage` if the kernel is small.
9. Type `make modules`
10. Type `make modules_install`
11. Copy the file `/usr/src/linux/arch/i386/boot/bzImage` to `/boot/vmlinuz` .
12. Copy the file `System.map` from `/usr/src/linux` to the `/boot` directory. Rename it `System.map=w.x.y` and make sure there is a link file called `System.map` pointing to it. This step will keep you from getting warning messages when you boot. It seems to be a step they forgot to include in the kernel-howto.
 - o `cp /usr/src/System.map /boot/System.map-2.2.14`
 - o `cd /boot`
 - o `rm System.map`
 - o `ln -s System.map-2.2.14 System.map`
13. Run `rdev` on the new kernel image to verify the root filesystem device. `rdev /boot/vmlinuz /dev/hda2` . Please note that this step may be optional dependent on whether you want to use the boot loader, lilo, to point to the root device.
14. Make sure the `/etc/lilo.conf` file is correct (`image=/boot/vmlinuz`), and run lilo by typing `lilo` . If you didn't set the root device in step 12, you will need a line like `root=/dev/hda2` in `/etc/lilo.conf` in the group of commands for your kernel. This will tell the kernel where your root filesystem is.
15. If your kernel has a feature supported by a module that is required to boot, you will need to make a RAM disk boot image or your system won't boot.
 - o Type `lsmod` and look to see if the `loop` module is loaded. If it is skip the next step.
 - o Type `insmod /lib/modules/2.2.14/block/loop.o`
Use this command if the loopback module is not installed. This assumes you compiled the support into your kernel. In my example, I used kernel version 2.2.14, but your kernel version may have a different number. Substitute the appropriate values.
 - o Type `mkinitrd /boot/initrd-2.2.14.img 2.2.14`
Again, this example is for kernel version 2.2.14. This command will create a RAM image module for your kernel to load into.
 - o Add an entry similar to `initrd=/boot/initrd-2.2.14.img` to the `/etc/lilo.conf` file for the stanza that is used to boot this kernel. An example of the stanza follows.
 - o `image=/boot/vmlinuz`
 - o `label=rhl`
 - o `initrd=/boot/initrd-2.2.14.img`
 - o `read-only`
 - o `root=/dev/hda2`

Read the section about LILO for more information or read the lilo and lilo.conf man pages.

16. Run lilo by typing "lilo". If you didn't set the root device in step 12, you will need a line like "root=/dev/hda2" in /etc/lilo.conf in the group of commands for your kernel. This will tell the kernel where your root filesystem is.

This step may be necessary if you are using a kernel previous to the 2.2 series. In the directory "/usr/include" may be several link files such as "asm", "linux", and "scsi". These are normally softlinks through "/usr/src/linux". If they are, you can direct the softlink file "usr/src/linux" to your new kernel file which may be in the form "linux-2.2.14". If you don't have the "linux" softlink file and don't want to create it, you will need to remove and redirect the softlink files in "/usr/include" to the proper location of your new kernel source.

How to determine what library a binary needs

Using the "ldd" command, type "ldd /bin/ls" to see the shared libraries used by the "ls" command. Do the same for any other binary you are interested in. This way if a binary (newly created, downloaded, etc) requires a library you can check your "lib" directory to see if it is there. If it is not, you will need to create and install it.

How to tell if a library is a.out or ELF type

Type "file library.so.x", substituting the name of your library for "library.so.x". If ELF is used the loader library "ld-linux.so" is required, and if a.out is used, "ld.so" is required.

How to find Kernel Modules

Modules can be found in "lib/modules/2.2.12-20" for kernel version 2.2.12-20. They are loadable modules ending in "*.o" that are used to support the kernel.

How To Make Linux Packages

Making a man page

1. Write a text file to be a manpage as outlined in the "LINUX MAN-PAGE-HOWTO". Many of the following sections should normally be considered mandatory: "NAME, SYNOPSIS, DESCRIPTION, OPTIONS, FILES, EXAMPLES, AUTHOR." The following sections may be optional: "ENVIRONMENT, DIAGNOSTICS, BUGS, SEE ALSO".
2. Modify the text file with macros that are used by various programs you will use to make and read man pages including groff and ghostscript? The macros are listed below(They are defined in the directory "/usr/lib/groff/tmac" and the normal file used is "tmac.an".):

.AL - Creates an automatic numbered list. ".AL A" creates A,B, C.. list.
.B - Bold
.BI
.BR - Bold Roman?
.DL - Creates a dashed list using dashes rather than bullets.
.DT - Tabs every half inch
.EL
.H - Heading, levels 1-7, 1 is highest. ".H 2" sets a level 2 heading
.HP

- .I - Italics
- .IB
- .IP
- .IR
- LI - Lists, creates a bulleted list
- .LE - Ends a bulleted list
- .LP
- .ML - Creates a list with the character of your choice.
- .TH - Title header
- .PD - Set distance between lines, normally "PD 10". "PD 0" sets no space
- .PP - Paragraph break?
- .RB
- .RE
- .RS
- .S
- .SB
- .SH - Section headers
- .SM
- .SS - Subsection header
- .TH - Title header
- .TP - Sets a new paragraph. Title Paragraph?
- .de - define a new macro (EX: ".de PP" define a macro PP). The commands follow this line. The last line "." indicates the end of the definition. This applies to the "tmac.an" file.
- .ft - Set font (EX: ".ft B" sets bold font. ".ft I" sets italics "\fB" sets bold)
- .in - Indent text (EX: ".in 0.90" indents text 9 tenths of an inch).
- .ll - Line length (EX ".ll 7I" sets the line length to seven inches)
- .nr - Number register (EX: ".nr H3 2" Restarts the numbering of third level headings at 2.
- .ps - Point size (EX: .ps 14 defines point size of 14)
- \s - Embedded point size command (EX: \s20 Changes point size to 20 within a sentence)
- .sp - Gives one blank line
- .ti - Temporary indent (EX: ".ti 2I" Temporarily indents a line 2 inches, the next line is back to normal)
- .vs - Vertical spacing (EX: .vs 14 sets vertical spacing to 14.)
- \fP - Set to the previous font
- \fR - Set font to Roman
- \fI - Set font to italics
- \fB - Set font to bold

3. Run groff on the text file. The file I made for my project "viewmod" starts as follows:

```
.TH VIEWMOD 1 "SEPTEMBER 1999" Linux "User Manuals"
.SH NAME
viewmod \- View or modify data in files or standard input and send results to stdout.
.SH SYNOPSIS
viewmod [inputfile] [option1] [option2]
.SH DESCRIPTION
Allows data from files or standard input to be viewed or modified with the results sent to
```

standard output. This program can operate on any type of files whether they are binary or text files. It was originally written to search files for hidden control characters and get rid of them along with conversion of DOS based text files to UNIX based text files and vice/versa. There are two option types listed below. They are display type option and line length option. Of the display type option, there are currently eight choices. Only one of these choices may be used at a time. Choices 0 through 4 were mainly intended to be used to view file contents, but could be used to create new files by redirecting standard output to a file. Choices 5 through 7 were intended to do file conversions to strip or add additional characters to a file. Display option 1 is the default, and the default line length is set to 80 characters.

.SH OPTIONS

.TP

.B \-D0

Sends all characters to standard output as though they are ASCII characters. Adds a LF if the line length will be exceeded.

The name of the text file is "viewmod.txt".

To run groff, type "groff -t -man -Tascii viewmod.txt > viewmod.1"

4. Copy the file "viewmod.1" to the subdirectory "/usr/man/man1". Note, this assumes that it is appropriate for this file to be put in the man 1 pages.

Appropriate locations to put man pages

- 1 - User commands that may be started by everyone.
- 2 - System calls, that is, functions provided by the kernel.
- 3 - Subroutines, that is, library functions.
- 4 - Devices, that is, special files in the /dev directory.
- 5 - File format descriptions, e.g. /etc/passwd.
- 6 - Games, self-explanatory.
- 7 - Miscellaneous, e.g. macro packages, conventions.
- 8 - System administration tools that only root can execute.
- 9 - Another (Linux specific) place for kernel routine documentation.
- n - New documentation, that may be moved to a more appropriate section.
- o - Old documentation, that may be kept for a grace period.
- l - Local documentation referring to this particular system.

Making makefiles

Dependencies:

The two lines below are an example of a dependency. The first line defines the dependency relationship, and the second line defines the command line. There may be several command lines.

```
test_example : this.o that.o
cc -o george this.o that.o
```

In the example the dependency relationship defines this.o and that.o are required. The program george is generated. Normally the program name is the same as the dependency label "test_example".

Make predefined macros:

- CC - The build in macro of the name of the compiler.
- \$@ - The current target
- \$\$@ - The current target, meaningful only on dependency lines. Not useable with GNU make.

- `$>` - Evaluates to the prerequisite that triggered the rule, such as `iodate.c` in a source to object suffix rule.
- `$?` – A list of prerequisites newer than the current target. Used to create libraries using `ar`.
- `::` - Allows a target to be specified multiple times. Can't specify it anytime with `!`. EX: `"libprocs ::video.c"` and `"libprocs :: test.c"`.

Below is an example Make file I created to compile and install a project with 3 source binaries (spooldata, stripchars, lmax) and 1 script file (printst) along with 3 man pages and 1 configuration file. NOTE: All commands must begin with the TAB character.

```
# Make file for the project myproject
SHELL = /bin/bash
BINLOC = /usr/local/tmp
CONFLOC = /usr/local/etc
MANLOC = /usr/man

EXEC1 = spooldata
EXEC2 = stripchars
EXEC3 = lmax

SHUTILS = printst
CONFFILES = spooldata.conf
MANFILES = stripchars.1 spooldata.8 spooldata.conf.5
PRUTILS = $(EXEC1) $(EXEC2) $(EXEC3)

OBS1 = $(EXEC1:=.o)
OBS2 = $(EXEC2:=.o)
OBS3 = $(EXEC3:=.o)

SRCS1 = $(EXEC1:=.c)
SRCS2 = $(EXEC2:=.c)
SRCS3 = $(EXEC3:=.c)

.SUFFIXES : .c .o
# CC = gcc

# $(PRUTILS): $(@.o)    #One way to do it, but won't work with GNU make
# $(CC) -o $@ $@.o

all: $(PRUTILS)    #invoked when "make all" typed.  Does make for EXEC1, EXEC2,
EXEC3.

$(EXEC1): $(OBS1)
$(CC) -o $(EXEC1) $(OBS1)

$(EXEC2): $(OBS2)
$(CC) -o $(EXEC2) $(OBS2)

$(EXEC3): $(OBS3)
$(CC) -o $(EXEC3) $(OBS3)

.c.o:                #Uses suffix rule to compile c file to get object
file
$(CC) -c $<

#$(OBS1): $(SRCS1)    #These lines are replaced by suffix rule above.
# $(CC) -c $(SRCS1)
#$(OBS2): $(SRCS2)
# $(CC) -c $(SRCS2)
#$(OBS3): $(SRCS3)
# $(CC) -c $(SRCS3)

install:
@if \                #creates directory if necessary
```

```

    [ ! -d $(BINLOC) ]; \
        then \
    mkdir $(BINLOC); \
fi

@for i in $(PRUTILS); do \      #Copies files in PRUTILS to their proper
location
    cp $$i $(BINLOC)/$$i; \
    chmod +x $(BINLOC)/$$i; \
    echo copying $$i to $(BINLOC)/$$i; \
done ; \

@for i in $(SHUTILS); do \
    cp $$i $(BINLOC)/$$i; \
    chmod +x $(BINLOC)/$$i; \
    echo copying $$i to $(BINLOC)/$$i; \
done

@if \
    [ ! -d $(CONFLOC) ]; \
then \
    mkdir $(CONFLOC); \
fi

@for i in $(CONFFILES); do \
    cp $$i $(CONFLOC)/$$i; \
    echo copying $$i to $(CONFLOC)/$$i; \
done

@for i in $(MANFILES); do \
    PAGENO=${i##*{i%[0-9]}}; \
    cp $$i $(MANLOC)/man$$PAGENO/$$i; \
    echo copying $$i to $(MANLOC)/man$$PAGENO/$$i; \
done

clean:
    rm -f *.o

```

Making install packages

tar cf myproject.tar /usr/local/myproject	Takes all files in myproject and puts them in new archive file.
tar rf myproject.tar ./thisfile	Adds thisfile to myproject.tar
gzip myproject.tar	Replaces myproject.tar with the file myproject.tar.gz
gzip -9 myproject.tar	Uses most compact compression, default is 6. Fastest is 1.

How to Build RPM packages

1. Zip the files into one tarred and gzipped file. Put all the files into one directory, then perform the zip. The files are placed in the directory myprint-1.0.0 for this example. Zip the files with the command:

```
tar cvfz myprint-1.0.0.tar.gz myprint-1.0.0
```

2. Copy the tarred and gzipped file to /usr/src/redhat/SOURCES.

```
cp myprint-1.0.0.tar.gz /usr/src/redhat/SOURCES
```

3. Make a spec file called myprint-1.0.0.spec similar to the one below:
 4. Summary: My print package
 5. Name: myprint
 6. Version: 1.0.0

```

7.   Release: 1
8.   Copyright: GPL
9.   Group: Development/Tools
10.  Source: /usr/local/myprint-1.0.0.tar.gz
11.
12.  Vendor: Mycompany
13.  Packager: George
14.
15.  %description
16.  This print package does everything I want it to do, including
    sending me a notice when files are printed.
17.  %prep
18.  %setup
19.  %build
20.  make all RPM_OPT_FLAGS="$RPM_OPT_FLAGS"
21.  %install
22.  make install $RPM_BUILD_ROOT
23.  %clean
24.  make clean $RPM_BUILD_ROOT
25.  %files
26.  %defattr(-,root,root)
27.  %config /usr/local/etc/spooldata.conf
28.  /usr/local/bin/stripchars
29.  /usr/local/bin/printst
30.  /usr/local/bin/spooldata
31.  /usr/local/bin/lmax
32.  /usr/man/man1/stripchars.1
33.  /usr/man/man1/printst.1
34.  /usr/man/man1/lmax.1
35.  /usr/man/man8/spooldata.8
36.  /usr/man/man5/spooldata.conf.5

```

37. Issue the command:

```
rpm -ba myprint-1.0.0.spec
```

A further explanation of the spec file:

- The source line:

Multiple files can exist in the source description such as:

- Source0: printst
- Source1: spooldata

It is easier to wrap everything into a tarred and zipped file as shown above.

- The Group description is required and tells where the program is to be placed in the RPM structure.

The groups are:

- Amusements/Games
- Amusements/Graphics
- Applications/Archiving
- Applications/Communications
- Applications/Databases
- Applications/Editors
- Applications/Emulators
- Applications/Engineering
- Applications/File
- Applications/Internet
- Applications/Multimedia Applications/Productivity
- Applications/Publishing
- Applications/System
- Applications/Text Development/Debuggers
- Development/Languages

- [Development/Libraries](#)
 - [Development/System Development/Tools](#)
 - [Documentation System](#)
 - [Environment/Base](#)
 - [System Environment/Daemons](#)
 - [System Environment/Kernel](#)
 - [System Environment/Libraries](#)
 - [System Environment/Shells](#)
 - [User Interface/Desktops](#)
 - [User Interface/X](#)
 - [User Interface/X](#)
 - [Hardware Support](#)
- **%prep**
This section is used to get the sources ready to build. There is a script macro used here called %setup that will do the prep automatically. If you would like to do it manually you may want to add lines similar to the following to this section:


```
rm -rf $RPM_BUILD_DIR/hbprint-1.3.0
tar xvzf $RPM_SOURCE_DIR/hbprint-1.3.0.tgz
```
 - **%build**
This section does not support macros as in the previous section. This section builds the executable files similar to a program compile. In the example, the Makefile has a target called "all" allowing it to build all sources so the following line is used for this section:


```
make all
```
 - **%install**
This section is used to install required files for the package to the proper locations on the system. This section is similar to the build section with commands required to do the install. Commands that copy the file to specific locations should be here. Since the make file includes an install target which copies required files to their respective locations, this section includes the following line:


```
make install
```
 - **%files**
This section is a listing of all files to be installed with a complete path to their final locations. This section and a full listing all files is required. This listing is used for the binary package.

Writing Linux Script Programs

An example script:

```
#!/bin/bash
#
```

```

# printst      This file manages print files in subdirectories
#              under the "/var/spool/spooldata" subdirectory.
#              It loops through each subdirectory, looking
#              for files that have the sticky bit set.  If it
#              is set, the file will be sent to the printer,
#              and the sticky bit will be cleared.
#              Files are sent to the chosen printer based on
#              the name of the directory and the configuration
#              file.
#
if [ -d /var/spool/spooldata ]; then # if this spool subdirectory exists
  for dfile in /var/spool/spooldata/*; do # for all files in spooldata
    if [ -d $dfile ]; then # if the file is a subdirectory
      fname=${dfile#/var/spool/spooldata/} # parameter expansion used to get filename
without path
      # get the printername from the configuration file based on the subdirectory
name
      prname=`grep -i ${fname} /usr/local/etc/spooldata.conf | cut -f2 -d" "`
      for ifile in $dfile/*; do # for all files in the subdirectory
        if [ -k $ifile ]; then # if the file has the sticky bit set
          lpr -P$prname $ifile # print the file
          chmod -t $ifile # clear the sticky bit
        fi
      done
    fi
  done
fi

```

Put this file in your cron schedule to be run about once per minute or encase it in the following statements

```

while [ 1 ]
do
  sleep 60
  .
  .
done

```

Be aware of several things in this file:

1. Where the variable prname is set, the output is directed with ` rather than '. Using ' will not work!!! This is documented in the bash man page under command expansion. \$(command) may also be used
2. Parameter expansion is used to extract the print name.
 - o The \$ is used to indicate parameter expansion.
 - o The { } are used to separate parameter expansion.
 - o The first string or variable is the parameter and the second one is the expansion element
 - o If a # is used to separate parameter from element, the part of the element not matching the beginning of the parameter is kept.
 - o If a % is used the part of the element not matching the end of the parameter is kept.
3. If you copy this file or write it using DOS or windows, be sure all carriage returns have been stripped from the script file or it will not run right. Linux does not use carriage returns in files and many programs will not operate correctly when running files that have carriage returns in them.

An example spooldata.conf file with the directory name in /var/spool listed first. The printer that the file is sent to is listed as the second argument on the line.

```

print1 lp1
print2 lp2
print3 lp3

```

A makefile script example

Scripting that checks a file and modifies it if required. Since it is from a makefile, this file is similar to a normal script but is slightly modified with lines that end with ";" characters. This section demonstrates modification of the rc.local file from a makefile so a script program, "printst", may be run in the background when the system starts. The variable "BINLOC" is defined earlier with the line "BINLOC = /usr/local/tmp". See the section on "Making Packages" under the header "Making makefiles".

```
# Add the path for $(BINLOC) to the /etc/rc.d/rc.local file
@PATHIN=`grep "path=" /etc/rc.d/rc.local | grep "$(BINLOC)"`; \ # See if the
BINLOC path is set in the rc.local file
HEADER=""; \
if \
  [ -z "$$PATHIN" ]; \
then \
  echo Adding $(BINLOC) to path in /etc/rc.d/rc.local; \
  echo >> /etc/rc.d/rc.local; \
  echo "##### Added by Print Service Install on `date` by `whoami`" >>
/etc/rc.d.rc.local; \
  HEADER=Y; \
  echo "export PATH=\$$PATH:$(BINLOC)" >> /etc/rc.d.rc.local; \
else \
  echo $(BINLOC) already set in path in file /etc/rc.d/rc.local; \
fi; \
PRINTST=`grep -i "printst &" /etc/rc.d/rc.local`; \ # See if the string printst
is in the rc.local file
if \
  [ -z "$$PRINTST" ]; \
then \
  echo Adding printst command to /etc/rc.d/rc.local file; \
  if \
    [ -z "$$HEADER" ]; \
  then \
    echo ↵> /etc/rc.d.rc.local; \
    echo "##### Added by Print Service Install on `date` by `whoami`" >>
/etc/rc.d.rc.local; \
    HEADER=Y; \
  fi; \
  echo "printst &" >> /etc/rc.d.rc.local; \
else \
  echo Already have printst command in /etc/rc.d/rc.local file; \
fi; \
if \
  [ -n "$$HEADER" ]; \
then \
  echo "##### End of Print Service Install Section #####" >>
/etc/rc.d.rc.local; \
  echo >> /etc/rc.d.rc.local; \
fi;
```

Running JAVA on Linux

Note in this series of instructions, you are told to recompile the kernel. This step may not be necessary on your system. If desired, you may skip recompiling the kernel, then if it does not work, recompile your kernel.

1. Copy the file "jdk1.2pre-v2.tar.bz2" from blackdown.org to "/usr/local".
2. Bunzip2 -kv jdk1.2pre-v2.tar.bz2

3. Type "tar -xvf jdk1.2pre-v2.tar". This will install the file to "/usr/local/jdk1.2". Rename it to /usr/local/java by typing "mv jdk1.2 java".
4. Get ready to re-compile the kernel. In "/usr/src/linux/fs/binfmt_java.c" modify "_PATH_JAVA" from "/usr/bin/java" to "/usr/local/java/bin/java" and the PATHAPPLET line to "/usr/local/java/bin/appletviewer"
5. Start kernel compilation by typing "cd /usr/src/linux" and "make config" and go through the configuration setup. Use the file in documentation for a guide. See the section in "The Linux Users Guide" on "Recompiling the Kernel".
6. Install the new kernel, and re-run lilo to set it up for use.
7. edit "/etc/profile, adding the path "/usr/local/java/bin" to the path statement.
8. Rename /usr/bin/java and /usr/bin/appletviewer to *.bak so they aren't run since they are in the path before the correct java path.
9. To run java type "java -cp ChatServer.jar ChatServer".

Linux Remote Booting a Diskless Computer

Remote booting a diskless computer involves using network services to load an operating system on a computer and running it. What this means is that the remote computer will have no need of a hard drive, floppy, CD-ROM or permanent storage media of any sort. It will only need its network card with a boot PROM on board in order to boot and run a remotely loaded operating system. Since there is no permanent storage media on the remote computer, it will need to store the operating system in its own RAM memory. Generally any programs that the user may want to run are also stored in RAM. However, the server can allow read access to the remote boot computer for accessing additional programs. Furthermore, depending on the boot mode, the user may be able to log onto the server and make changes to files they have permission to access. This will in no way compromise the security of the server and access to the server can be controlled easily by administrators. This is explained in more detail later.

Having diskless computers offers several advantages and disadvantages:

Advantages:

1. Lower cost of most computers fielded.
2. Remote computers can be configured from one location.
 1. Future upgrades and changes to the remote computers' configuration are easy to make.
 2. The administrator can choose the operating system the remote computer will boot from the server.
3. Users can use remote computers without worrying about messing up their configuration.
4. Outdated computers from 386 based micro's and above can be used.

Disadvantages:

1. If the server for remote booting fails, the remote computers won't function, but redundancy can solve this.
2. The administrators must keep track of Remote computer NIC hardware address(MAC) to configure computers.

Hardware Requirements:

1. 386 based microprocessor or better, 486 recommended.
2. 8M RAM minimum, 16 or more recommended.
3. Network Card.

Operating Systems that can currently be run on the remote computer include Linux and DOS. Linux can be run in a terminal mode or X windows mode depending on the configuration set by the server. X windows mode is a graphical user mode similar to Microsoft Windows 95/98/NT/2000/etc.

There are several network services that must be provided by the server computer which are as follows:

1. BOOTP - Allows the remote computer to get an IP address when it boots. This lets it identify itself and start using the other network services to continue its boot process. The IP address is assigned by the server based on the hardware(MAC) address of its network interface card(NIC).
2. TFTP - Allows the remote computer to load an image file that contains the operating system
3. NFS - Network File System, Allows the remote computer to have read access to a shared file system on the server computer in order to support Linux booting.

Miscellaneous

How to set time

While logged in as root do the following:

1. Type "date".
2. You should see some variation of"

"Wed Nov 24, 9:29:17 EST 1999"

3. To change the time type(as an example):

```
date -s 10:10
```

4. The system response will be:

"Wed Nov 24, 10:10:02 EST 1999"

5. Then if you want to set the hardware(BIOS) clock so the system will keep the time when it reboots type:

```
clock -w
```

or

```
setclock
```

The program setclock will set your hardware clock based on your system configuration parameters including whether or not your clock is set to universal time.

How to paste text in files

There is a cut and paste mouse utility that works with virtual consoles called gpm which runs as a daemon. To use it,

1. Move your mouse to the text you want to cut or paste
2. Hold the left mouse button down
3. Drag the mouse to the end of the selected text
4. Release the mouse button
5. If deleting text, just press the "DEL" key for your final step. If pasting text, move the text cursor to the location you want to paste to by switching terminals with function keys, using arrow keys, etc.
6. If pasting, press the right mouse button.

How to re-display previously displayed text

Text that has scrolled off the top of the screen may be viewed again using the <SHIFT><PgUp> key combination. The Keys in the numbers section on the far right of the keypad do not work for this function, only the grey PgUp and PgDn keys just to the right of the <Enter> key. If you want other keys to perform this function, it would be necessary to map them for bash shell keymapping. Pressing any other key other than <SHIFT><PgUp or ><SHIFT><PgDn> will bring you back to the normal screen location.

How to set up cron commands

The system administrator can schedule tasks by adding entries to the /etc/crontab (see crontab(5)) file or on Redhat Linux, by adding entries in one of the cron.hourly, cron.daily, cron.weekly, or cron.monthly files. Users may be able to schedule cron jobs if the system is configured to allow it. If neither of the /etc/cron.allow nor the /etc/cron.deny files exist, either all users will be able to run cron commands or no users will be able to do it. If /etc/cron.allow exists, the user must be listed their in order to use cron commands. If /etc/cron.deny exists, the user must not be listed here or they will be unable to use cron.

One useful entry you can put in the /etc/cron.weekly directory in a file named something like "cleanold.cron" is:

```
find /var/spool/myspools -mtime +33 -exec rm -f {} \;
```

This entry will remove all files in the /var/spool/myspools directory whose data was changed more than 33 days ago.

The user (if allowed) can schedule cron tasks by following the below procedure:

1. Make a crontab file called for example "mycron".
2. Use the crontab(1) command to submit the command(s) to cron by typing "crontab mycron".
3. You can view what you have installed by typing "crontab -l".

crontab commands:

crontab -e	Starts vi session by default to edit your crontab file. To use emacs as your editor, type "export VISUAL=emacs" before typing this command. When you exit the editor, the modified crontab is installed automatically.
crontab -r	Removes your crontab entry from the /var/spool/cron directory. Does not erase your original crontab file.
crontab -l	Lists all the user's cron tasks.

To use emacs, type "export VISUAL=emacs" before starting crontab.

How to start Linux from DOS

Use the DOS program "loadlin", usually with a Linux distribution on the CDROM under /dosutils/loadlin.