DRAGONFLY BSD – A CASE STUDY

Aaditya Umashankar- CB.EN.U4CSE18402
Sahasra Bommineni – CB.EN.U4CSE18414
Phanindra Reddy - CB.EN.U4CSE18433
Shri Hari Nithin K M - CB.EN.U4CSE18456
Swaran Karthikeyan - CB.EN.U4CSE18461
Vijay Swaminathan - CB.EN.U4CSE18467

INTRODUCTION

DragonFly BSD is a free and open source Unix like operating system forked from FreeBSD 4.8. Matthew Dillon, an Amiga developer in the late 1980s and early 1990s and FreeBSD developer between 1994 and 2003, began working on DragonFly BSD in June 2003 and announced it on the FreeBSD mailing lists on 16 July 2003.

Dillon started DragonFly in the belief that the techniques adopted for threading and symmetric multiprocessing in FreeBSD 5. would lead to poor performance and maintenance problems.

He sought to correct these anticipated problems within the FreeBSD project. Due to conflicts with other FreeBSD developers over the implementation of his ideas, his ability to directly change the codebase was eventually revoked.

Despite this, the DragonFly BSD and FreeBSD projects still work together, sharing bug fixes, driver updates, and other improvements.

FEATURES

Education:

Are you a student of computer science or a related engineering field? There is no better way of learning about operating systems, computer architecture, and networking than the hands-on, under-the-hood experience that DragonFly can provide.

Research:

With source code for the entire system available, DragonFly is an excellent platform for research in operating systems as well as other branches of computer science.

Networking:

Need a new router? A name server (DNS)? A firewall to keep people out of your internal network? DragonFly can easily turn that unused older PC sitting in the corner into an advanced router with sophisticated packet-filtering capabilities.

X Window workstation:

Unlike an X terminal, DragonFly allows many applications to be run locally if desired, thus relieving the burden on a central server. DragonFly can even boot *diskless*, making individual workstations even cheaper and easier to administer.

Software Development:

The basic DragonFly system comes with a full complement of development tools including the renowned GNU C/C++ compiler and debugger.

INSTALLATION PROCEDURE FOR DRAGONFLY BSD OS

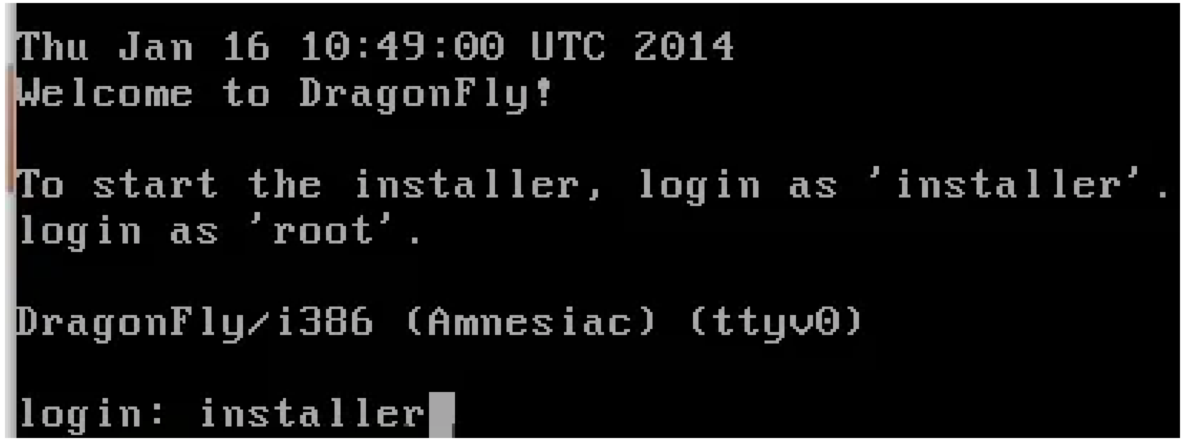
Create a virtual machine in VirtualBox using the FreeBSD (64 bit) template with 4 GB memory, 128 GB dynamically allocated VDI virtual disk image

Download and extract the x86_64 CD installation media and then add it to the virtual machine, but remember to remove it before taking VirtualBox snapshots.

Start the virtual machine. Before it finishes booting, the will be greeted with the following boot menu.

After the computer finishes booting, you will be greeted with the following welcome message and login prompt.

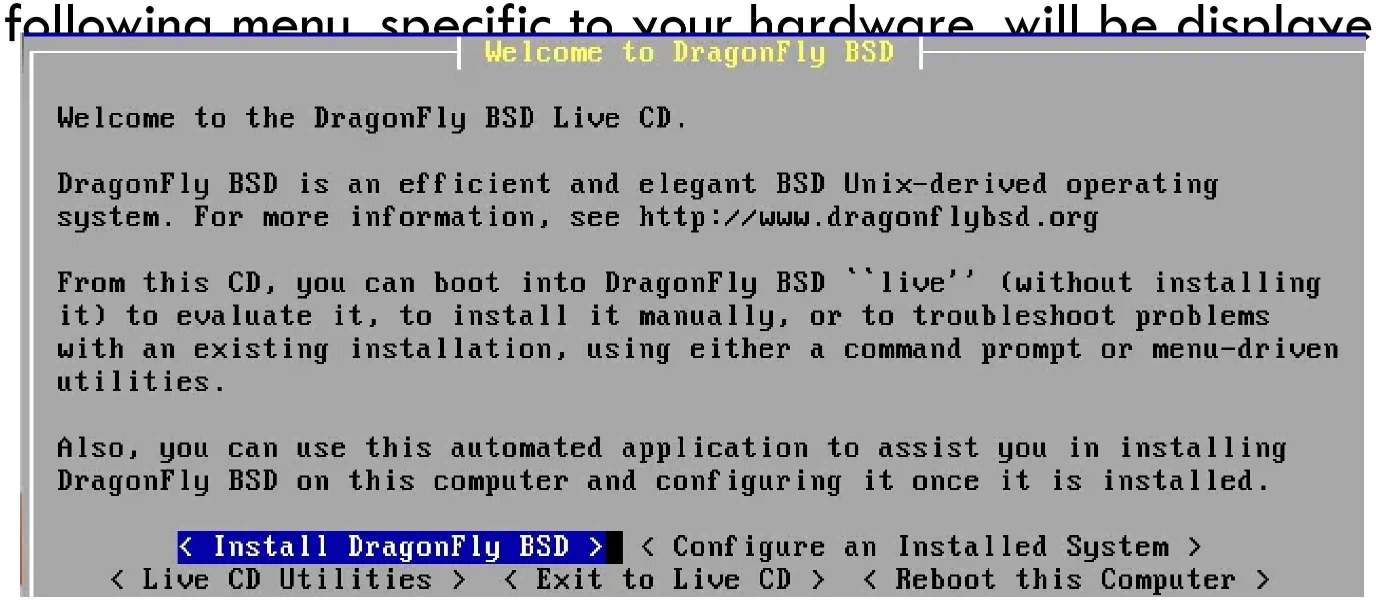
Type 'installer' without quotes and press enter.



```
Thu Jan 16 10:49:00 UTC 2014
Welcome to DragonFly!

To start the installer, login as 'installer'.
login as 'root'.

DragonFly/i386 (Amnesiac) (ttyv0)

login: installer
```

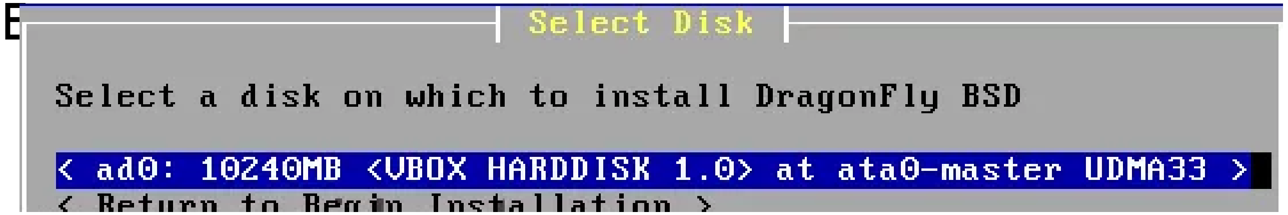
Highlight the option < Install DragonFly BSD > and press enter.

The following menu, specific to your hardware, will be displayed

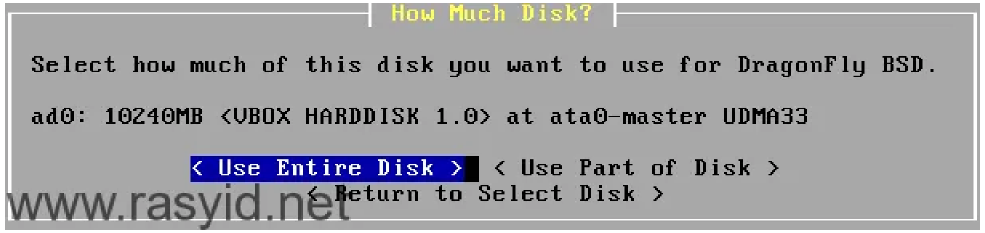


```
                 Welcome to DragonFly BSD

Welcome to the DragonFly BSD Live CD.

DragonFly BSD is an efficient and elegant BSD Unix-derived operating
system. For more information, see http://www.dragonflybsd.org

From this CD, you can boot into DragonFly BSD ``live'' (without installing
it) to evaluate it, to install it manually, or to troubleshoot problems
with an existing installation, using either a command prompt or menu-driven
utilities.

Also, you can use this automated application to assist you in installing
DragonFly BSD on this computer and configuring it once it is installed.

      < Install DragonFly BSD > < Configure an Installed System >
   < Live CD Utilities > < Exit to Live CD > < Reboot this Computer >
```

Highlight the disk on which to install DragonFly BSD, in this case ad0, and press E

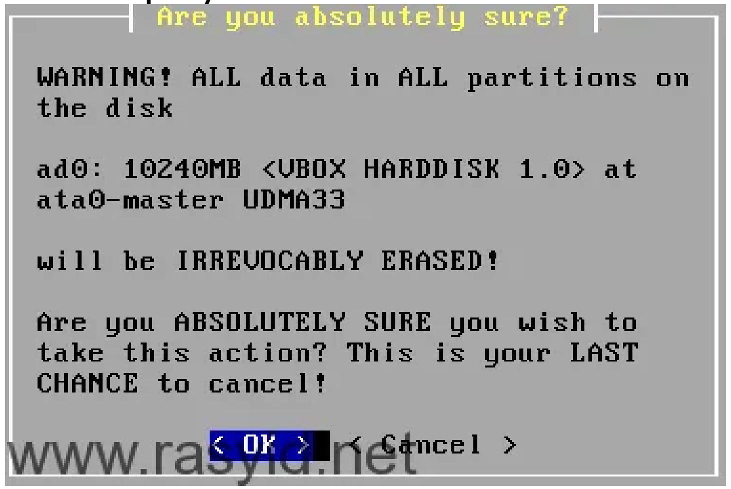


```
                     Select Disk

 Select a disk on which to install DragonFly BSD

 < ad0: 10240MB <VBOX HARDDISK 1.0> at ata0-master UDMA33 >
 < Return to Begin Installation >
```



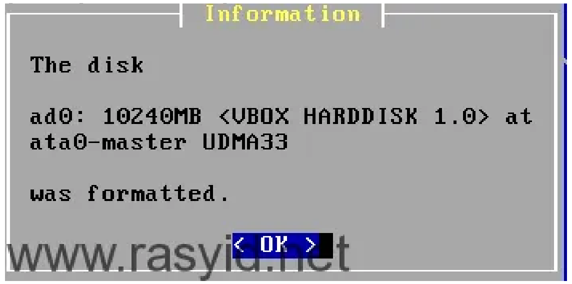
```
How Much Disk?

Select how much of this disk you want to use for DragonFly BSD.

ad0: 10240MB <VBOX HARDDISK 1.0> at ata0-master UDMA33

            < Use Entire Disk >  < Use Part of Disk >
                   < Return to Select Disk >
```

- If you would like to use your entire disk, then highlight < Use Entire Disk > and press enter.
- The following menu will be displayed.



```
Are you absolutely sure?

WARNING! ALL data in ALL partitions on
the disk

ad0: 10240MB <VBOX HARDDISK 1.0> at
ata0-master UDMA33

will be IRREVOCABLY ERASED!

Are you ABSOLUTELY SURE you wish to
take this action? This is your LAST
CHANCE to cancel!

            < OK >  < Cancel >
```

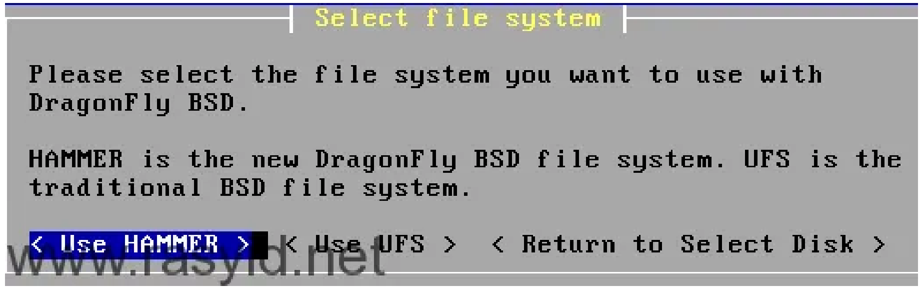
- If you are absolutely sure, highlight < OK > and press enter.
- The following menu will



```
Information

The disk

ad0: 10240MB <VBOX HARDDISK 1.0> at
ata0-master UDMA33

was formatted.

              < OK >
```

Press enter.

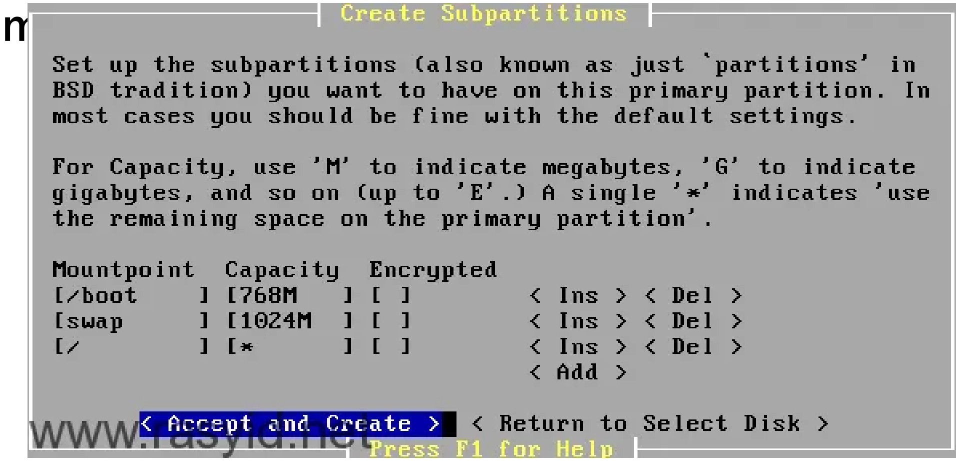
The following menu will be displayed



```
Select file system

Please select the file system you want to use with
DragonFly BSD.

HAMMER is the new DragonFly BSD file system. UFS is the
traditional BSD file system.

< Use HAMMER > < Use UFS >  < Return to Select Disk >
```

If you want to use HAMMER, then highlight HAMMER and press enter.

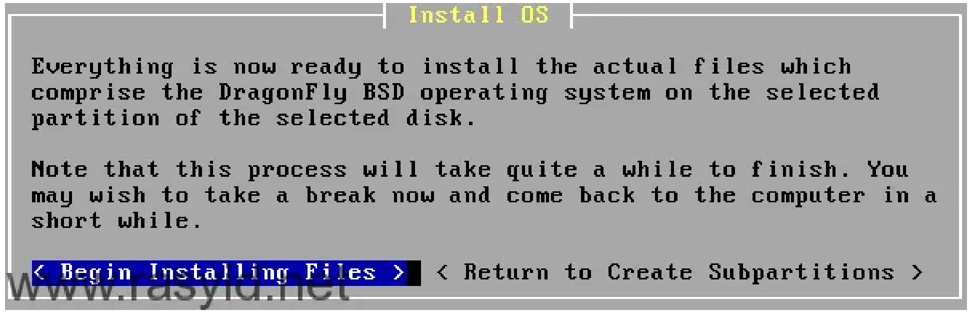
The following m



```
Create Subpartitions

Set up the subpartitions (also known as just `partitions' in
BSD tradition) you want to have on this primary partition. In
most cases you should be fine with the default settings.

For Capacity, use 'M' to indicate megabytes, 'G' to indicate
gigabytes, and so on (up to 'E'.) A single '*' indicates 'use
the remaining space on the primary partition'.

Mountpoint  Capacity  Encrypted
[/boot    ] [768M   ] [ ]        < Ins > < Del >
[swap     ] [1024M  ] [ ]        < Ins > < Del >
[/        ] [*      ] [ ]        < Ins > < Del >
                                 < Add >

< Accept and Create > < Return to Select Disk >
Press F1 for Help
```

Highlight < Accept and Create > and press enter.

The following menu will be displayed



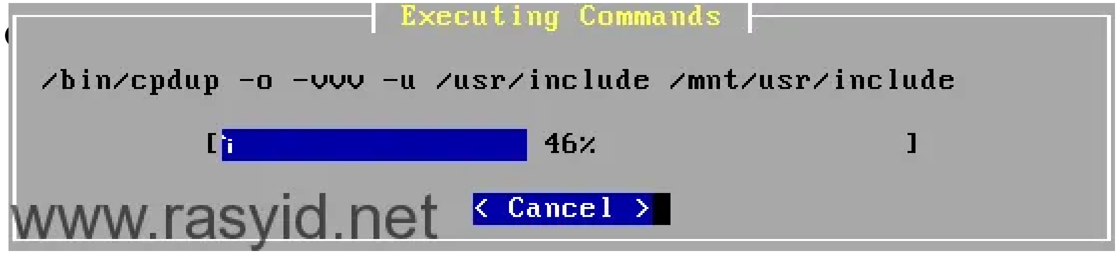
Everything is now ready to install the actual files which comprise the DragonFly BSD operating system on the selected partition of the selected disk.

Note that this process will take quite a while to finish. You may wish to take a break now and come back to the computer in a short while.

< Begin Installing Files >
< Return to Create Subpartitions >

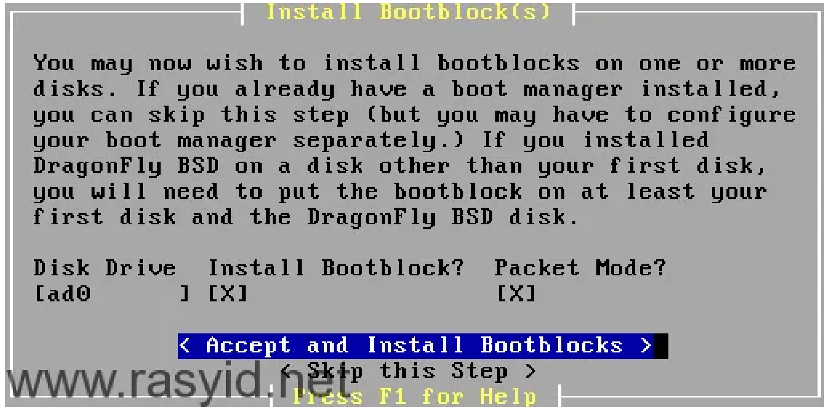
Highlight < Begin Installing Files > and press enter.

The foll



```
/bin/cpdup -o -vvv -u /usr/include /mnt/usr/include
```

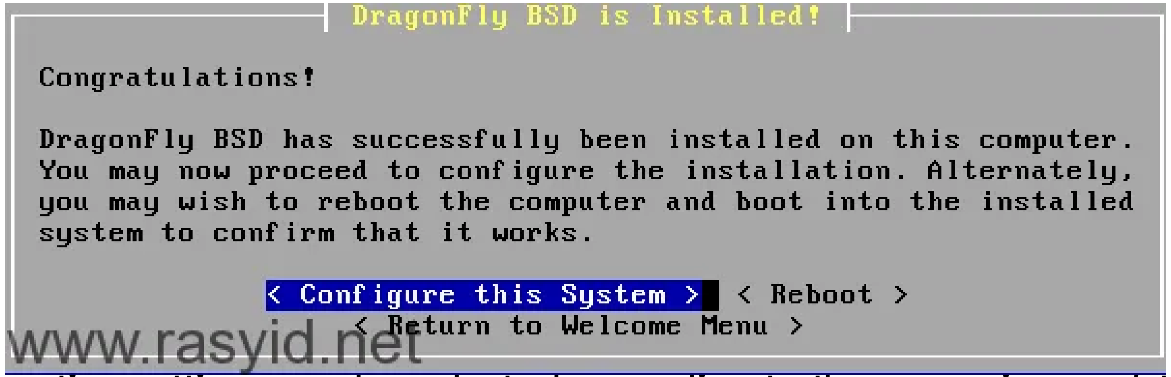
When it finishes, the following menu will be displayed



```
                Install Bootblock(s)

You may now wish to install bootblocks on one or more
disks. If you already have a boot manager installed,
you can skip this step (but you may have to configure
your boot manager separately.) If you installed
DragonFly BSD on a disk other than your first disk,
you will need to put the bootblock on at least your
first disk and the DragonFly BSD disk.

Disk Drive   Install Bootblock?   Packet Mode?
[ad0       ] [X]                  [X]

            < Accept and Install Bootblocks >
                  < Skip this Step >
                  Press F1 for Help
```

Highlight < Accept and Install Bootblocks > and press enter.

The following dialog box will be displayed.



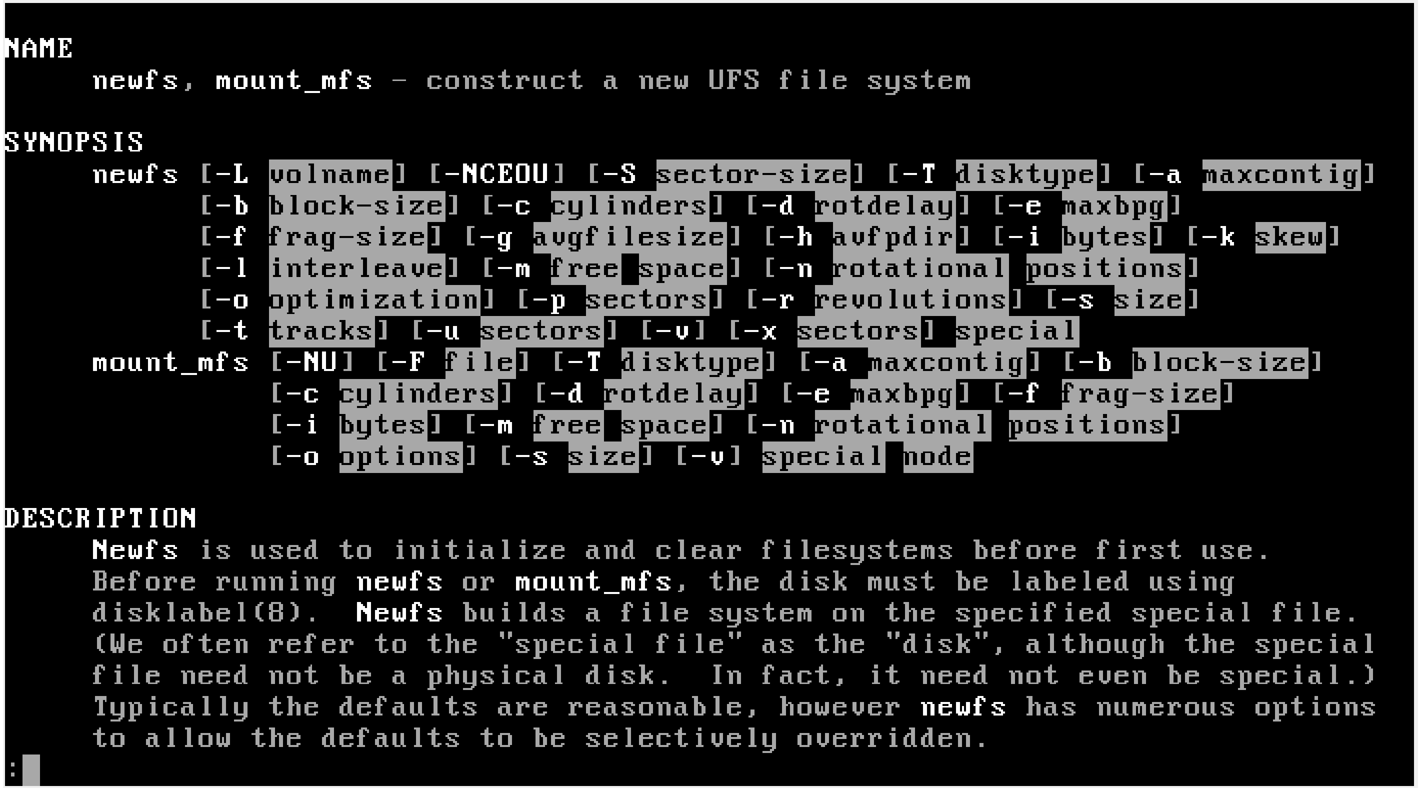
```
              DragonFly BSD is Installed!

Congratulations!

DragonFly BSD has successfully been installed on this computer.
You may now proceed to configure the installation. Alternately,
you may wish to reboot the computer and boot into the installed
system to confirm that it works.

          < Configure this System > < Reboot >
              < Return to Welcome Menu >
```

The configuration settings can be selected according to the convenience of the user.

COMMANDS IN DRAGONFLY BSD OS

Newfs

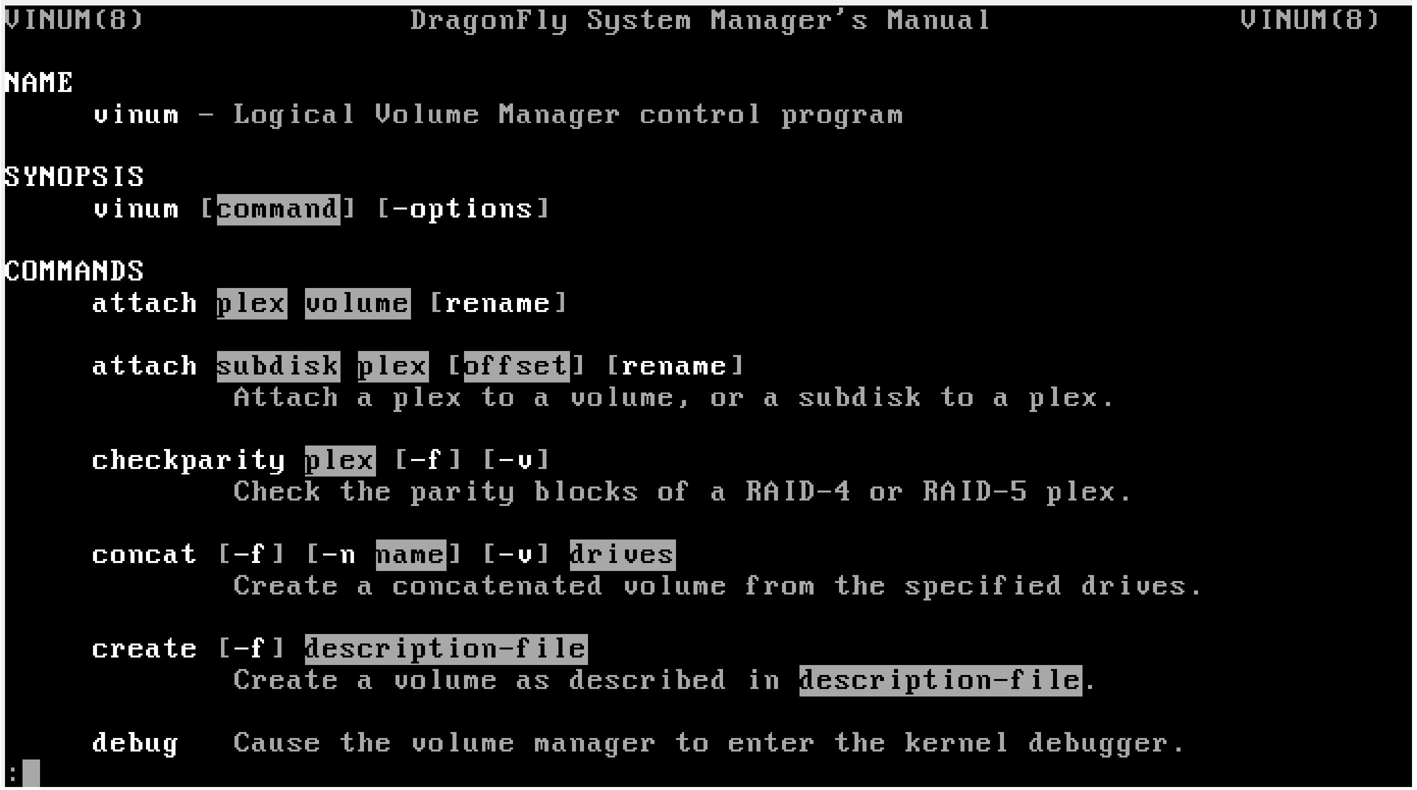
- construct a new UFS file system



```
NAME
     newfs, mount_mfs - construct a new UFS file system

SYNOPSIS
     newfs [-L volname] [-NCEOU] [-S sector-size] [-T disktype] [-a maxcontig]
           [-b block-size] [-c cylinders] [-d rotdelay] [-e maxbpg]
           [-f frag-size] [-g avgfilesize] [-h avfpdir] [-i bytes] [-k skew]
           [-l interleave] [-m free space] [-n rotational positions]
           [-o optimization] [-p sectors] [-r revolutions] [-s size]
           [-t tracks] [-u sectors] [-v] [-x sectors] special
     mount_mfs [-NU] [-F file] [-T disktype] [-a maxcontig] [-b block-size]
               [-c cylinders] [-d rotdelay] [-e maxbpg] [-f frag-size]
               [-i bytes] [-m free space] [-n rotational positions]
               [-o options] [-s size] [-v] special node

DESCRIPTION
     Newfs is used to initialize and clear filesystems before first use.
     Before running newfs or mount_mfs, the disk must be labeled using
     disklabel(8).  Newfs builds a file system on the specified special file.
     (We often refer to the "special file" as the "disk", although the special
     file need not be a physical disk.  In fact, it need not even be special.)
     Typically the defaults are reasonable, however newfs has numerous options
     to allow the defaults to be selectively overridden.
:
```

Vinum

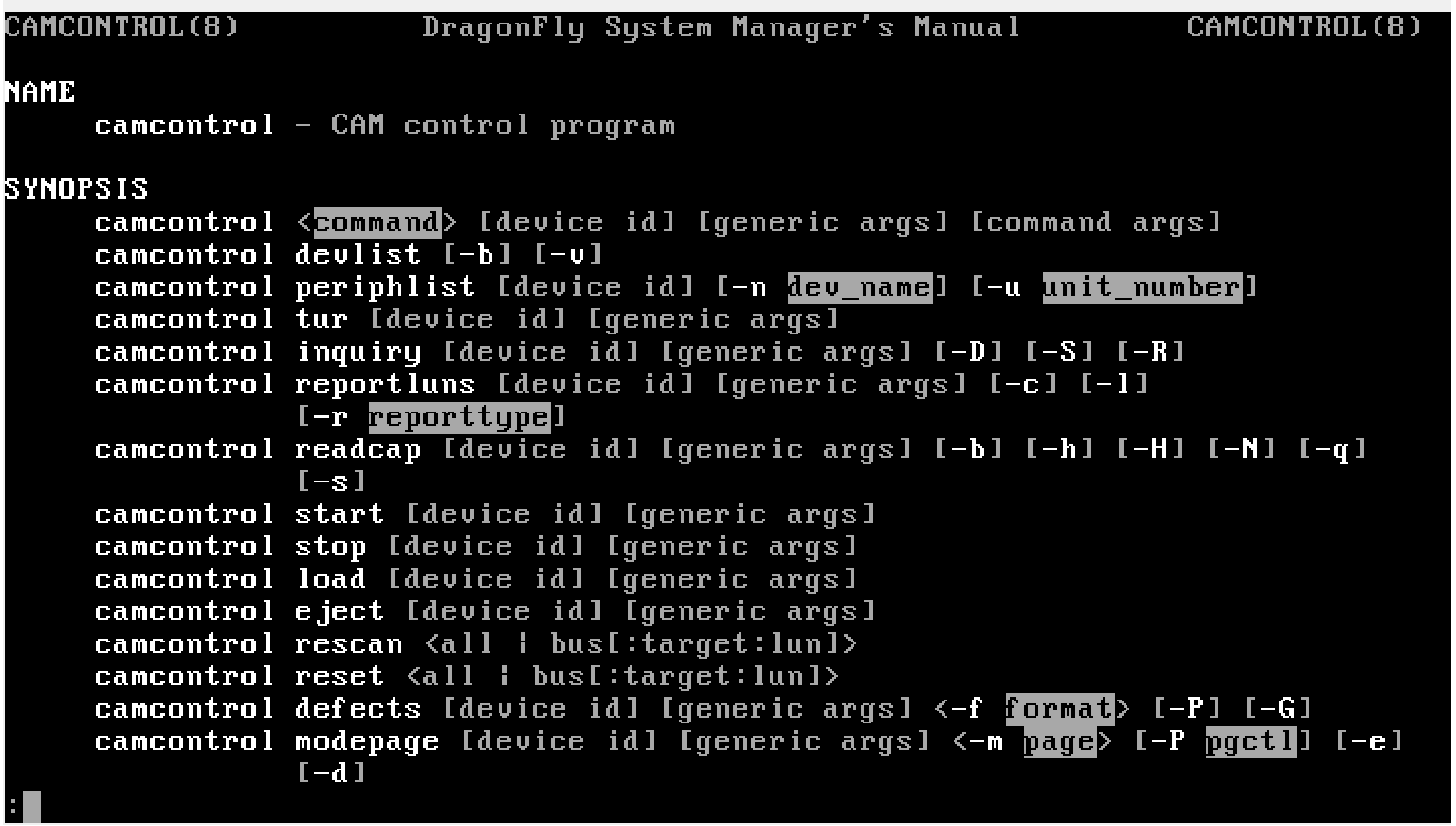
- Logical Volume Manager control program



```
VINUM(8)                DragonFly System Manager's Manual               VINUM(8)

NAME
     vinum - Logical Volume Manager control program

SYNOPSIS
     vinum [command] [-options]

COMMANDS
     attach plex volume [rename]

     attach subdisk plex [offset] [rename]
             Attach a plex to a volume, or a subdisk to a plex.

     checkparity plex [-f] [-v]
             Check the parity blocks of a RAID-4 or RAID-5 plex.

     concat [-f] [-n name] [-v] drives
             Create a concatenated volume from the specified drives.

     create [-f] description-file
             Create a volume as described in description-file.

     debug   Cause the volume manager to enter the kernel debugger.
:
```

Camcontrol

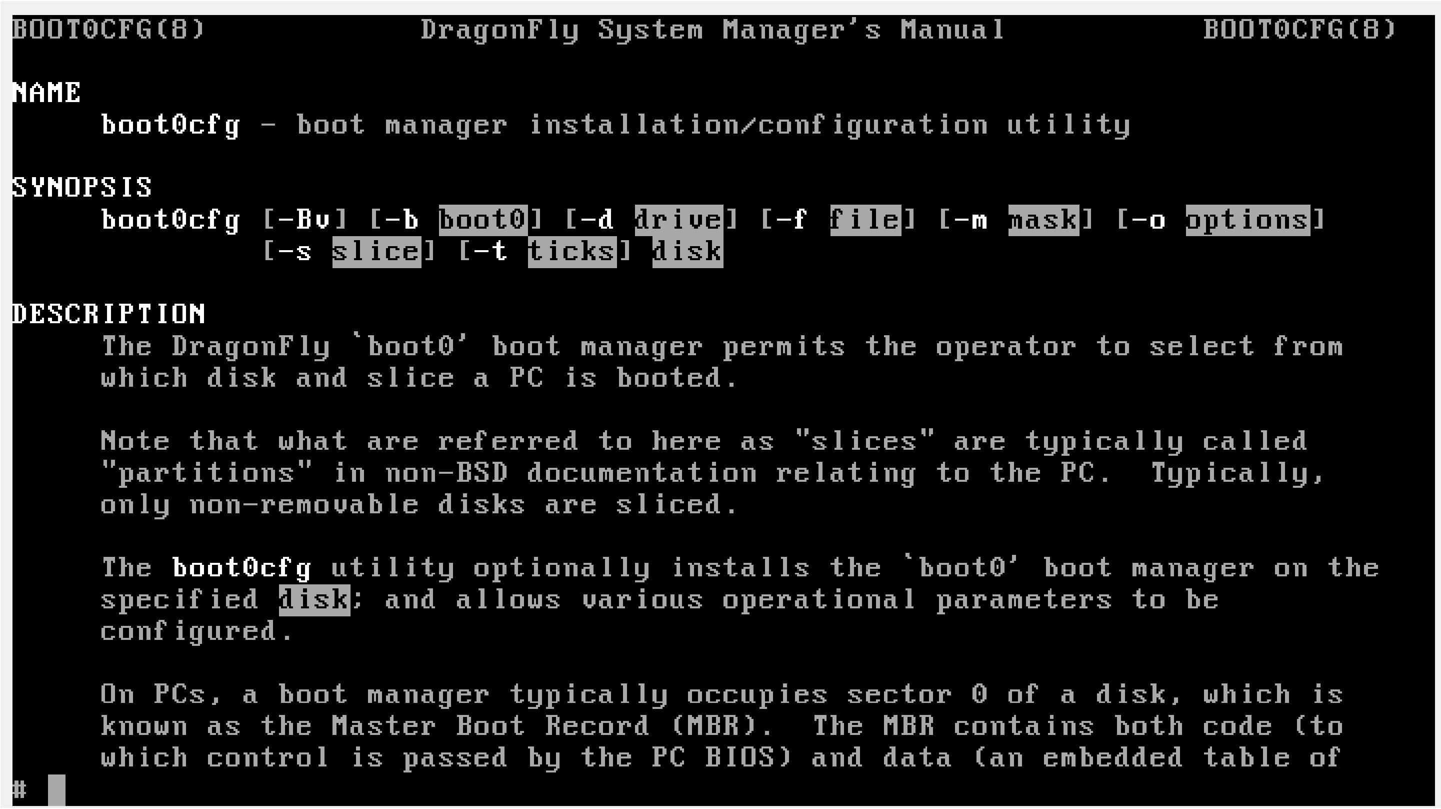
- CAM control program



```
CAMCONTROL(8)          DragonFly System Manager's Manual          CAMCONTROL(8)

NAME
     camcontrol – CAM control program

SYNOPSIS
     camcontrol <command> [device id] [generic args] [command args]
     camcontrol devlist [-b] [-v]
     camcontrol periphlist [device id] [-n dev_name] [-u unit_number]
     camcontrol tur [device id] [generic args]
     camcontrol inquiry [device id] [generic args] [-D] [-S] [-R]
     camcontrol reportluns [device id] [generic args] [-c] [-l]
                [-r reporttype]
     camcontrol readcap [device id] [generic args] [-b] [-h] [-H] [-N] [-q]
                [-s]
     camcontrol start [device id] [generic args]
     camcontrol stop [device id] [generic args]
     camcontrol load [device id] [generic args]
     camcontrol eject [device id] [generic args]
     camcontrol rescan <all | bus[:target:lun]>
     camcontrol reset <all | bus[:target:lun]>
     camcontrol defects [device id] [generic args] <-f format> [-P] [-G]
     camcontrol modepage [device id] [generic args] <-m page> [-P pgctl] [-e]
                [-d]
:
```

Boot0cfg

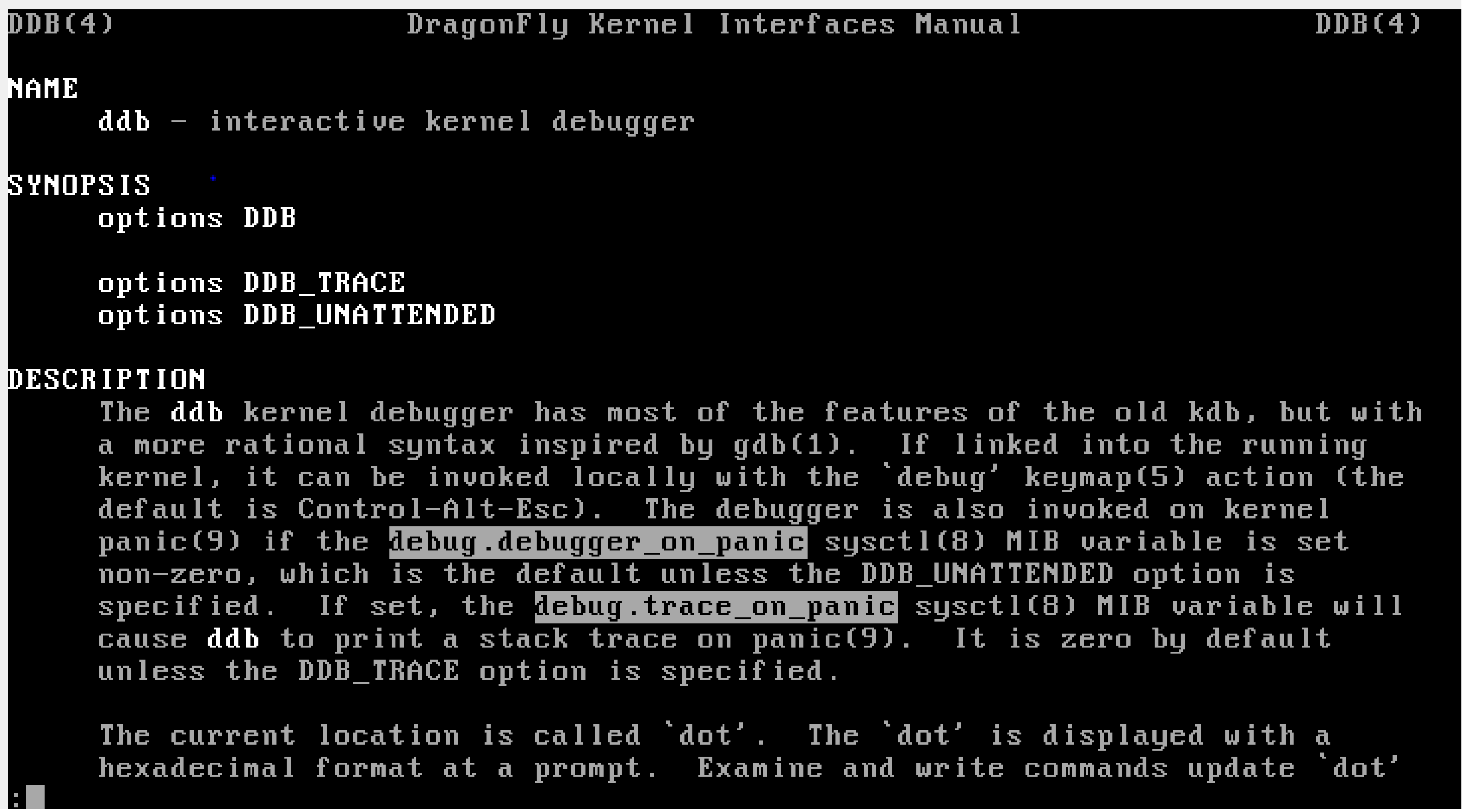
- boot manager installation/configuration utility



```
BOOT0CFG(8)            DragonFly System Manager's Manual            BOOT0CFG(8)

NAME
     boot0cfg - boot manager installation/configuration utility

SYNOPSIS
     boot0cfg [-Bv] [-b boot0] [-d drive] [-f file] [-m mask] [-o options]
              [-s slice] [-t ticks] disk

DESCRIPTION
     The DragonFly `boot0' boot manager permits the operator to select from
     which disk and slice a PC is booted.

     Note that what are referred to here as "slices" are typically called
     "partitions" in non-BSD documentation relating to the PC.  Typically,
     only non-removable disks are sliced.

     The boot0cfg utility optionally installs the `boot0' boot manager on the
     specified disk; and allows various operational parameters to be
     configured.

     On PCs, a boot manager typically occupies sector 0 of a disk, which is
     known as the Master Boot Record (MBR).  The MBR contains both code (to
     which control is passed by the PC BIOS) and data (an embedded table of
#
```

Ddb

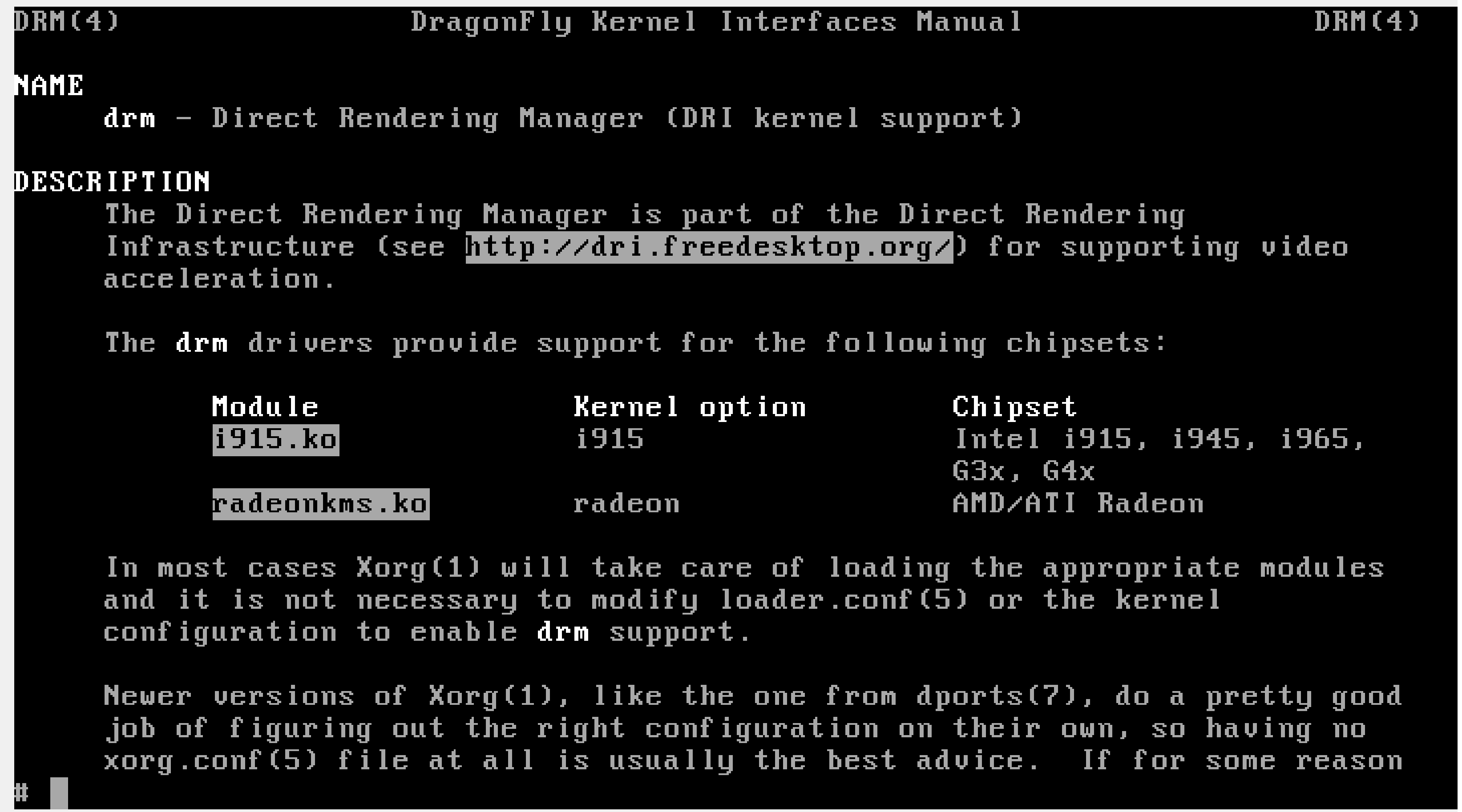
- interactive kernel debugger



```
DDB(4)                 DragonFly Kernel Interfaces Manual                 DDB(4)

NAME
     ddb - interactive kernel debugger

SYNOPSIS
     options DDB

     options DDB_TRACE
     options DDB_UNATTENDED

DESCRIPTION
     The ddb kernel debugger has most of the features of the old kdb, but with
     a more rational syntax inspired by gdb(1).  If linked into the running
     kernel, it can be invoked locally with the `debug' keymap(5) action (the
     default is Control-Alt-Esc).  The debugger is also invoked on kernel
     panic(9) if the debug.debugger_on_panic sysctl(8) MIB variable is set
     non-zero, which is the default unless the DDB_UNATTENDED option is
     specified.  If set, the debug.trace_on_panic sysctl(8) MIB variable will
     cause ddb to print a stack trace on panic(9).  It is zero by default
     unless the DDB_TRACE option is specified.

     The current location is called `dot'.  The `dot' is displayed with a
     hexadecimal format at a prompt.  Examine and write commands update `dot'
:
```

Drm

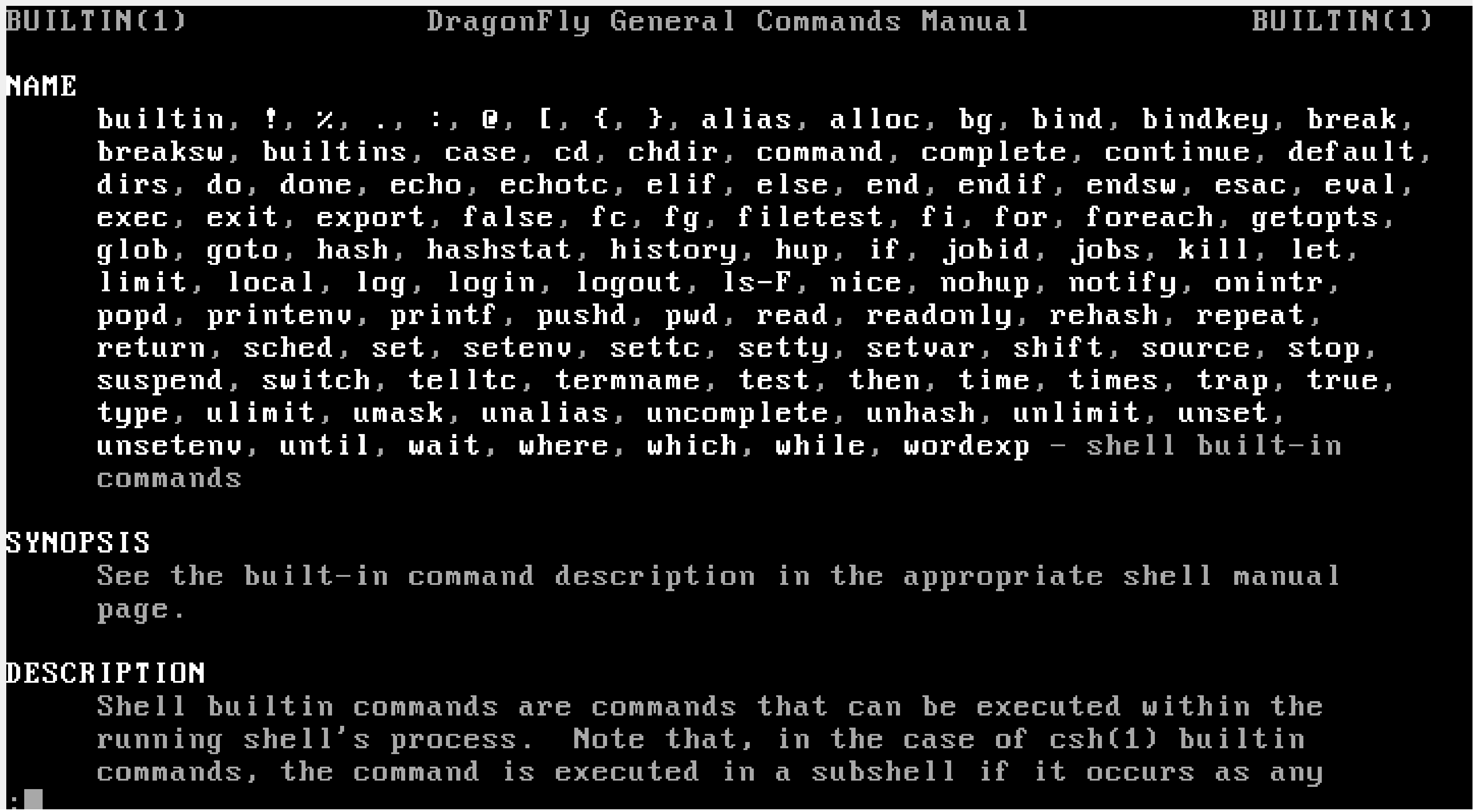
- Direct Rendering Manager (DRI kernel support)



```
DRM(4)                DragonFly Kernel Interfaces Manual                DRM(4)

NAME
     drm - Direct Rendering Manager (DRI kernel support)

DESCRIPTION
     The Direct Rendering Manager is part of the Direct Rendering
     Infrastructure (see http://dri.freedesktop.org/) for supporting video
     acceleration.

     The drm drivers provide support for the following chipsets:

           Module              Kernel option         Chipset
           i915.ko             i915                  Intel i915, i945, i965,
                                                     G3x, G4x

           radeonkms.ko        radeon                AMD/ATI Radeon

     In most cases Xorg(1) will take care of loading the appropriate modules
     and it is not necessary to modify loader.conf(5) or the kernel
     configuration to enable drm support.

     Newer versions of Xorg(1), like the one from dports(7), do a pretty good
     job of figuring out the right configuration on their own, so having no
     xorg.conf(5) file at all is usually the best advice.  If for some reason
#
```

Unset

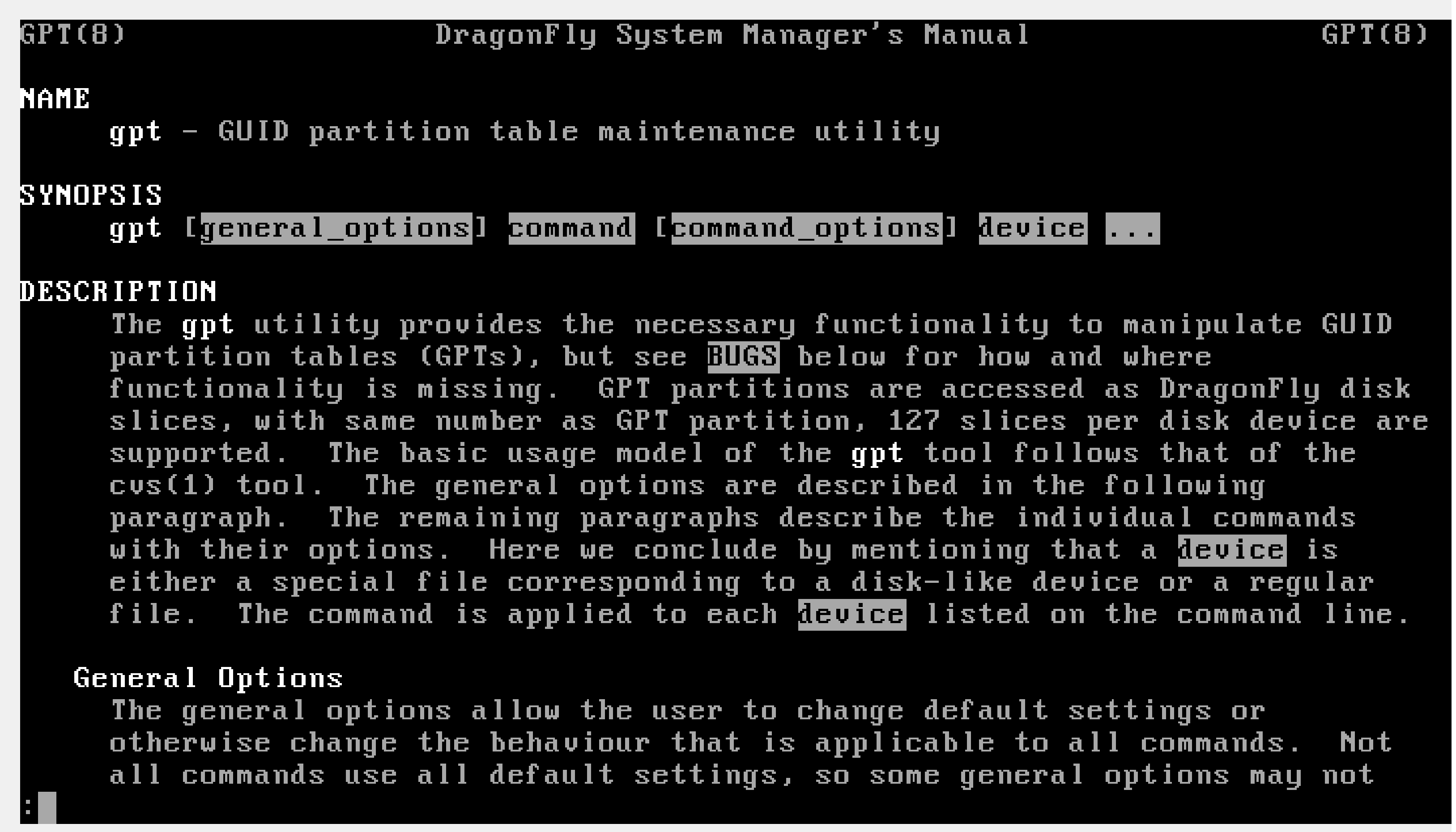
- to clear local environment variable



```
BUILTIN(1)             DragonFly General Commands Manual             BUILTIN(1)

NAME
     builtin, !, %, ., :, @, [, {, }, alias, alloc, bg, bind, bindkey, break,
     breaksw, builtins, case, cd, chdir, command, complete, continue, default,
     dirs, do, done, echo, echotc, elif, else, end, endif, endsw, esac, eval,
     exec, exit, export, false, fc, fg, filetest, fi, for, foreach, getopts,
     glob, goto, hash, hashstat, history, hup, if, jobid, jobs, kill, let,
     limit, local, log, login, logout, ls-F, nice, nohup, notify, onintr,
     popd, printenv, printf, pushd, pwd, read, readonly, rehash, repeat,
     return, sched, set, setenv, settc, setty, setvar, shift, source, stop,
     suspend, switch, telltc, termname, test, then, time, times, trap, true,
     type, ulimit, umask, unalias, uncomplete, unhash, unlimit, unset,
     unsetenv, until, wait, where, which, while, wordexp - shell built-in
     commands

SYNOPSIS
     See the built-in command description in the appropriate shell manual
     page.

DESCRIPTION
     Shell builtin commands are commands that can be executed within the
     running shell's process.  Note that, in the case of csh(1) builtin
     commands, the command is executed in a subshell if it occurs as any
:
```

Gpt

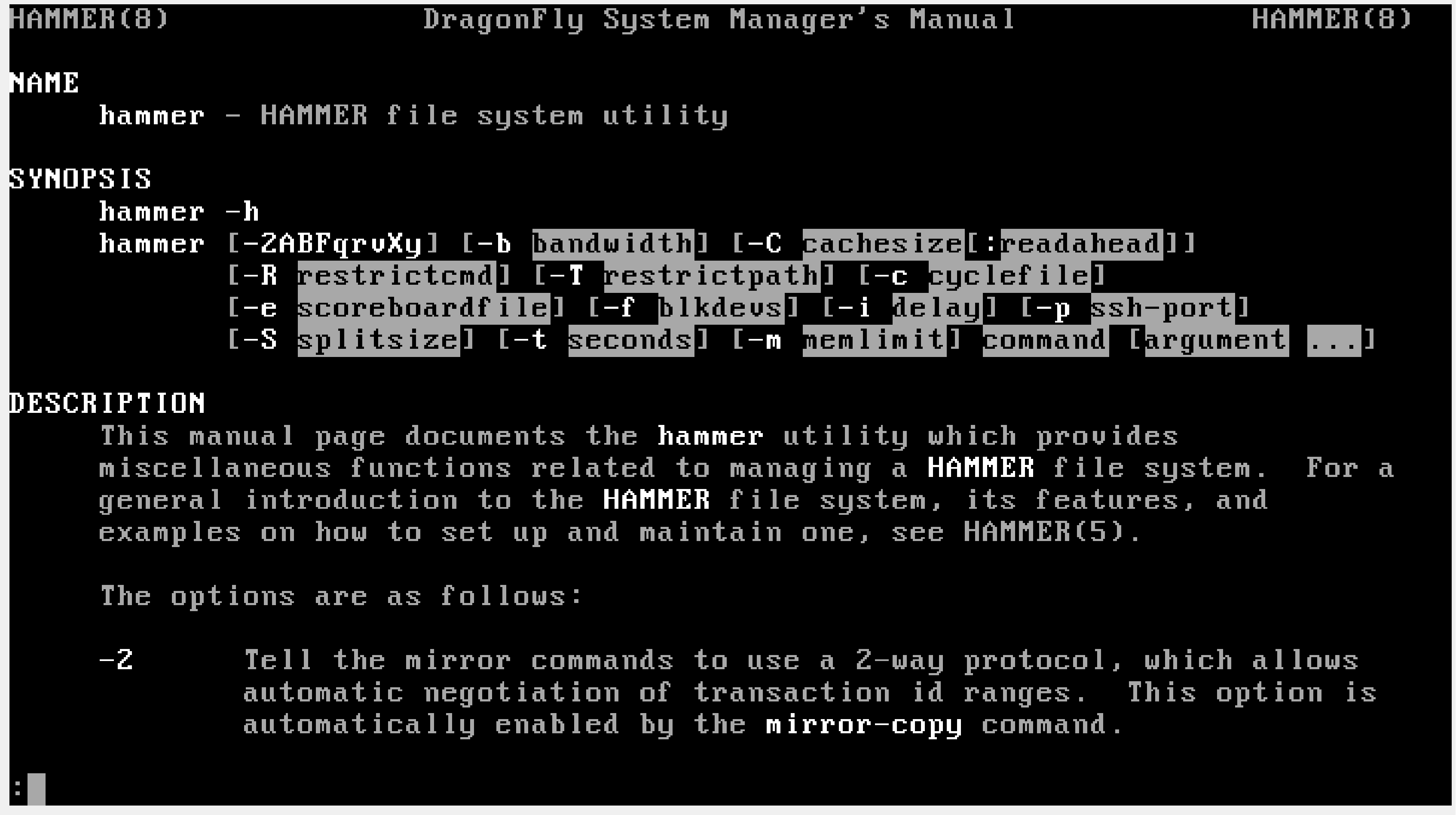
- GUID partition table maintenance utility



```
GPT(8)                  DragonFly System Manager's Manual                  GPT(8)

NAME
     gpt - GUID partition table maintenance utility

SYNOPSIS
     gpt [general_options] command [command_options] device ...

DESCRIPTION
     The gpt utility provides the necessary functionality to manipulate GUID
     partition tables (GPTs), but see BUGS below for how and where
     functionality is missing.  GPT partitions are accessed as DragonFly disk
     slices, with same number as GPT partition, 127 slices per disk device are
     supported.  The basic usage model of the gpt tool follows that of the
     cvs(1) tool.  The general options are described in the following
     paragraph.  The remaining paragraphs describe the individual commands
     with their options.  Here we conclude by mentioning that a device is
     either a special file corresponding to a disk-like device or a regular
     file.  The command is applied to each device listed on the command line.

   General Options
     The general options allow the user to change default settings or
     otherwise change the behaviour that is applicable to all commands.  Not
     all commands use all default settings, so some general options may not
:
```

Hammer

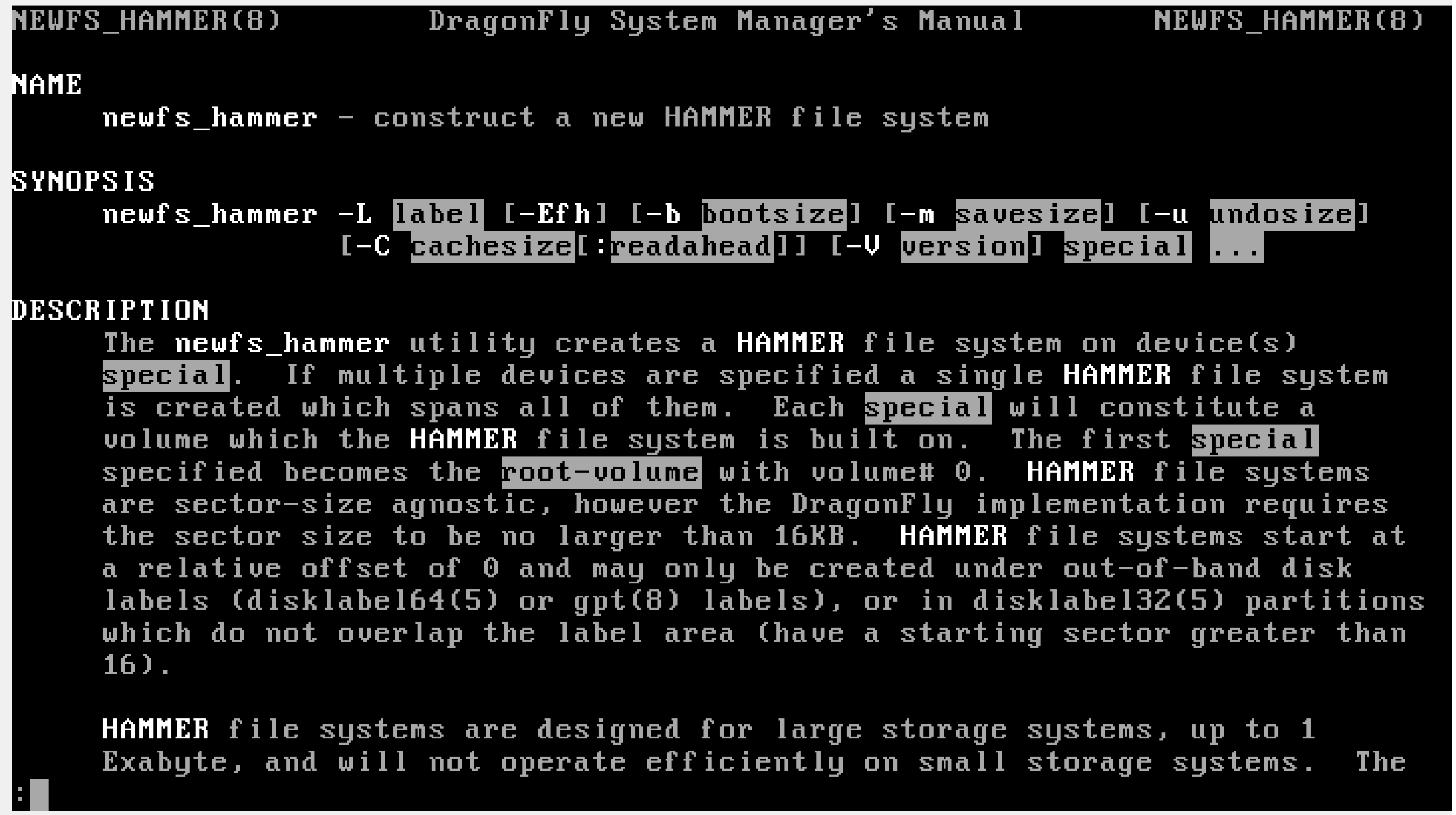
- HAMMER file system utility



```
HAMMER(8)               DragonFly System Manager's Manual               HAMMER(8)

NAME
     hammer - HAMMER file system utility

SYNOPSIS
     hammer -h
     hammer [-2ABFqrvXy] [-b bandwidth] [-C cachesize[:readahead]]
            [-R restrictcmd] [-T restrictpath] [-c cyclefile]
            [-e scoreboardfile] [-f blkdevs] [-i delay] [-p ssh-port]
            [-S splitsize] [-t seconds] [-m memlimit] command [argument ...]

DESCRIPTION
     This manual page documents the hammer utility which provides
     miscellaneous functions related to managing a HAMMER file system.  For a
     general introduction to the HAMMER file system, its features, and
     examples on how to set up and maintain one, see HAMMER(5).

     The options are as follows:

     -2       Tell the mirror commands to use a 2-way protocol, which allows
              automatic negotiation of transaction id ranges.  This option is
              automatically enabled by the mirror-copy command.

:
```

Newfs_hammer

- construct a new HAMMER file system



```
NEWFS_HAMMER(8)        DragonFly System Manager's Manual       NEWFS_HAMMER(8)

NAME
     newfs_hammer - construct a new HAMMER file system

SYNOPSIS
     newfs_hammer -L label [-Efh] [-b bootsize] [-m savesize] [-u undosize]
                  [-C cachesize[:readahead]] [-V version] special ...

DESCRIPTION
     The newfs_hammer utility creates a HAMMER file system on device(s)
     special.  If multiple devices are specified a single HAMMER file system
     is created which spans all of them.  Each special will constitute a
     volume which the HAMMER file system is built on.  The first special
     specified becomes the root-volume with volume# 0.  HAMMER file systems
     are sector-size agnostic, however the DragonFly implementation requires
     the sector size to be no larger than 16KB.  HAMMER file systems start at
     a relative offset of 0 and may only be created under out-of-band disk
     labels (disklabel64(5) or gpt(8) labels), or in disklabel32(5) partitions
     which do not overlap the label area (have a starting sector greater than
     16).

     HAMMER file systems are designed for large storage systems, up to 1
     Exabyte, and will not operate efficiently on small storage systems.  The
:
```

System Calls in DragonFly BSD OS

Process System Calls

procctl - control reaping of sub-processes

- Description
  - The procctl() system call allows a process to take-over the reaping task from init for any forked sub-process, recursively (for all children thereafter) which would otherwise reparent to init. This allows a chain of control to be maintained no matter what the sub-process does.
  - Any process may become a reaper for its sub-processes. The feature may also be used recursively, or independently, to create reaping domains or sub-domains. This call is typically used by service monitoring programs, jails, or chroots to ensure that the underlying services cannot get away from under the monitor.
- Syntax
  - procctl(idtype_t idtype, id_t id, int cmd, void *data);

Rexec - return stream to a remote command

- Description
  - The rexec() function looks up the host *ahost using gethostbyname(3), returning -1 if the host does not exist. Otherwise *ahost is set to the standard name of the host. If a username and password are both specified, then these are used to authenticate to the foreign host; otherwise the environment and then the user's .netrc file in his home directory are searched for appropriate information. If all this fails, the user is prompted for the information.
- Syntax
  - rexec(char **ahost, int inport, char *user, char *passwd, char *cmd, int *fd2p);

execl, execlp, execle, exect, execv, execvp, execvP -- execute a file

- Description
  - The exec family of functions replaces the current process image with a new process image. The functions described in this manual page are front-ends for the function execve(2). (See the manual page for execve(2) for detailed information about the replacement of the current process. The script(7) manual page provides detailed information about the execution of interpreter scripts. The initial argument for these functions is the pathname of a file which is to be executed.
- Syntax
  - Int execv(const char *path, char *const argv[]);

Rfork - manipulate process resources

- Description
  - Forking, vforking or rforking are the only ways new processes are created. The flags argument to rfork() selects which resources of the invoking process (parent) are shared by the new process (child) or initialized to their default values. The resources include the open file descriptor table (which, when shared, permits processes to open and close files for other processes), and open files. Flags is the logical OR of some subset of:
    - RFPROC
    - RFNOWAIT
    - RFFDG
    - RFCFDG
    - RFMEM
    - RFSIGSHARE
    - RFLINUXTHPN
  - Syntax
    - rfork(int flags);

stat, readlink -- display file status

- Description
  - The stat utility displays information about the file pointed to by file.Read, write, or execute permissions of the named file are not required, but all directories listed in the pathname leading to the file must be searchable. If no argument is given, stat displays information about the file descriptor for standard input. When invoked as readlink, only the target of the symbolic link is printed. If the given argument is not a symbolic link and the -f option is not specified, readlink will print nothing and exit with an error. If the -f option is specified, the output is canonicalized by following every symlink in every component of the given path recursively. Readlink will resolve both absolute and relative paths, and return the absolute pathname corresponding to file. In this case, the argument does not need to be a symbolic link.
- Syntax
  - stat [-FHLnq] [-f format | -l | -r | -s | -x] [-t timefmt] [file ...] readlink [-fn] [file ...]

read, readv, pread, preadv, extpread, extpreadv - read input

- Description
  - Read() attempts to read nbytes of data from the object referenced by the descriptor d into the buffer pointed to by buf. Readv() performs the same action, but scatters the input data into the iovcnt buffers specified by the members of the iov array: iov[0], iov[1], ..., iov[iovcnt-1]. The pread(), preadv(), extpread(), and extpreadv() calls perform the same function, but read from the specified position in the file without modifying the file pointer.
- Syntax
  - ssize_t pread(int d, void *buf, size_t nbytes, off_t offset)

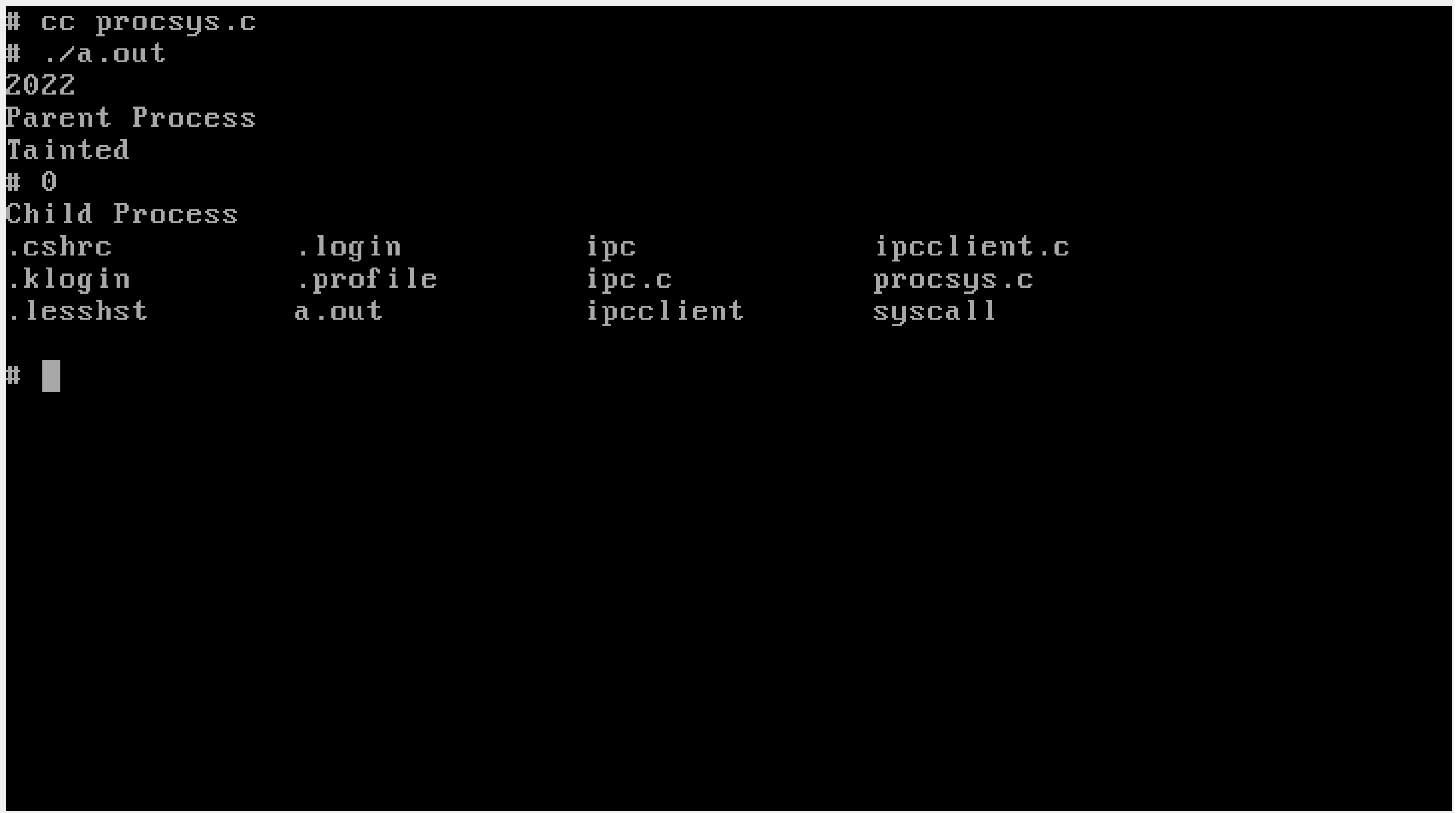
write, writev, pwrite, pwritev, extpwrite, extpwritev -- write output

- Description
  - Write() attempts to write nbytes of data to the object referenced by the descriptor d from the buffer pointed to by buf. Writev() and pwritev() perform the same action, but gather the output data from the iovcnt buffers specified by the members of the iov array: iov[0], iov[1], ..., iov[iovcnt-1]. Pwrite() and pwritev() perform the same function, but write to the specified position in the file without modifying the file pointer.
- Syntax
  - ssize_t pwrite(int d, const void *buf, size_t nbytes, off_t offset);

C PROGRAM FOR PROCESS BASED SYSTEM CALL

```
#include<stdio.h> #include<unistd.h> #include<stdlib.h>

int main(int argc,char *argv[]){

        int pid=rfork(RFPROC);

        printf(“%d”,pid);

        if(pid==0){

                printf(“\nParent Process\n”);

                exexcv(“/bin/ls”,argv);

        }

        else

                printf(“\n Parent Process”);

        if(issetupugid()==0)

                printf(“\nTainted\n”);

        else

                printf(“\nNot Tainted\n”);

return 0;
```

OUTPUT FOR PREVIOUS CODE



```
# cc procsys.c
# ./a.out
2022
Parent Process
Tainted
# 0
Child Process
.cshrc          .login          ipc             ipcclient.c
.klogin         .profile        ipc.c           procsys.c
.lesshst        a.out           ipcclient       syscall

#
```

INTERPROCESS COMMUNICATION USING SHARED MEMORY

Linux:

- Shmget: Flag can have the following value:
  - IPC_CREAT |0666
  - IPC_EXCL |0666
- Shmat: Flag can have the following value:
  - SHM_RND
  - SHM_EXEC
  - SHM_RDONLY
  - SHM_REMAP

Dragonfly BSD:

- Shmget: Flag can have the following value:
  - IPC_CREAT |SHM_R |SHM_W|(SHM_R>>3)|(SHM_W>>3)|(SHM_R>>6)|(SHM_W>>6)
  - IPC_EXCL |SHM_R |SHM_W|(SHM_R>>3)|(SHM_W>>3)|(SHM_R>>6)|(SHM_W>>6)
- Shmat: Flag can have only SHM_RND as value.

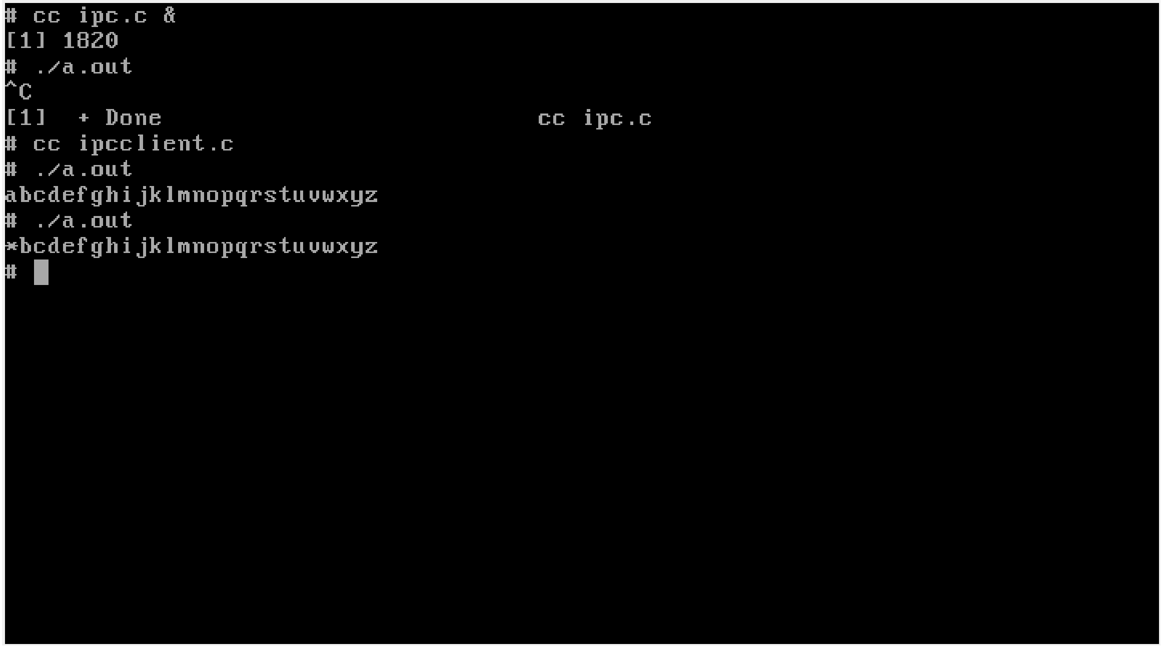
PROGRAM FOR IPC USING SHARED MEMORY(SERVER)

```
#include <sys/types.h> #include <sys/ipc.h> #include <sys/shm.h> #include <stdio.h> #include <stdlib.h>#define MAXSIZE     27

void die(char *s){

    perror(s);

    exit(1);

}

int main(){

    char c;

    int shmid;

    key_t key;

    char *shm, *s;

    key = 5678;

    if ((shmid = shmget(key, MAXSIZE, IPC_CREAT |SHM_R |SHM_W|(SHM_R>>3) |(SHM_W>>3)|(SHM_R>>6)|(SHM_W>>6 )) < 0)

       die("shmget");

    if ((shm = shmat(shmid, NULL, 0)) == (char *) -1)

        die("shmat");

    s = shm;

    for (c = 'a'; c <= 'z'; c++)

        *s++ = c;

    while (*shm != '*')
```

PROGRAM FOR IPC USING SHARED MEMORY(CLIENT)

```
#include <sys/types.h>#include <sys/ipc.h>#include <sys/shm.h>#include <stdio.h>#include <stdlib.h>#define MAXSIZE     27
void die(char *s){
    perror(s);
    exit(1); }
int main(){
    int shmid;
    key_t key;
    char *shm, *s;
    key = 5678;
    if((shmid = shmget(key, MAXSIZE,SHM_R |SHM_W|(SHM_R>>3) |(SHM_W>>3)|(SHM_R>>6)|(SHM_W>>6 ) )) < 0)
       die("shmget");
if ((shm = shmat(shmid, NULL, 0)) == (char *) -1)
        die("shmat");
for (s = shm; *s != '\0'; s++)
        putchar(*s);
 putchar('\n');
    *shm = '*';
```

OUTPUT OF THE PREVIOUS CODE



```
# cc ipc.c &
[1] 1820
# ./a.out
^C
[1]  + Done                          cc ipc.c
# cc ipcclient.c
# ./a.out
abcdefghijklmnopqrstuvwxyz
# ./a.out
*bcdefghijklmnopqrstuvwxyz
# 
```