Using GRUB

The GRUB homepage can be found at http://www.gnu.org/software/grub/grub.en.html.

This tutorial was written by Chris Giese and is also availible at his site.

```
================================================================
References
================================================================
This information was gleaned from my own experiments with GRUB
and from the following posts to alt.os.development:

Subject: Re: generic bootloader question
Newsgroups: alt.os.development
From: "Marv"
Date: Sat, 7 Apr 2001 23:35:20 +0100
References: <9antu8$glc$1@uni00nw.unity.ncsu.edu>
Message-ID: <986682856.680474@dionysos>

Subject: Re: Grub multiboot example
Newsgroups: alt.os.development
From: "Marv"
Date: Mon, 4 Jun 2001 17:21:17 +0100
References: <4a400d54.0106040458.5140872b@posting.google.com>
Message-ID:

Subject: Re: Grub multiboot example
Newsgroups: alt.os.development
From: "Mike Wimpy"
Date: Thu, 7 Jun 2001 22:17:51 -0700
References: <4a400d54.0106040458.5140872b@posting.google.com> <3B1CDA6D.154ADD9D@127.0.0.1> <3B1CDAF9
Message-ID: <3b205ff8_2@news.pacifier.com>

Subject: Re: grub coff (solved it!)
Newsgroups: alt.os.development
From: "Mark & Candice White"
Date: Sun, 16 Sep 2001 10:57:34 GMT
References:
Message-ID:

================================================================
Getting GRUB
================================================================
Source code:
        ftp://alpha.gnu.org/gnu/grub/grub-0.90.tar.gz

Binaries:
        ftp://alpha.gnu.org/gnu/grub/grub-0.90-i386-pc.tar.gz

DOS and Windows users will need PARTCOPY or RAWRITE:
        http://www.execpc.com/~geezer/johnfine/index.htm#zero
        http://uranus.it.swin.edu.au/~jn/linux/rawwrite.htm
        http://www.tux.org/pub/dos/rawrite/

================================================================
Building GRUB
================================================================
UNIX
----
        configure ; make
(...I think...)

DOS and Windows
---------------
(ha! forget it)

================================================================
Installing GRUB on a floppy with no filesystem
================================================================
```

1. Get the GRUB binaries (files "stage1" and "stage2")

2. Concatenate the files "stage1" and "stage2" into one file:

```
(DOS)   copy /b stage1 + stage2 boot

(UNIX)  cat stage1 stage2 >boot
```

3. Write the file "boot" directly to the floppy disk:

```
(DOS)   rawrite boot a:
            -OR-
(DOS)   partcopy boot 0 168000 -f0

(UNIX)  cat boot >/dev/fd0
```

PARTCOPY will give an error message because the file "boot" is much shorter than 0x168000 bytes, but this is OK.

Installing GRUB on a floppy with a filesystem

1. Make a bootable GRUB floppy with no filesystem, as described above.

2. Copy the files "stage1" and "stage2" to a second floppy disk, one formatted with a filesystem that GRUB recognizes. To use the GRUB "setup" command, these files must be stored in subdirectory "/boot/grub":

```
(DOS)   mkdir a:\boot
        mkdir a:\boot\grub
        copy stage1 a:\boot\grub
        copy stage2 a:\boot\grub

(UNIX)  mount /dev/fd0 /mnt
        mkdir /mnt/boot
        mkdir /mnt/boot/grub
        cp stage1 /mnt/boot/grub
        cp stage2 /mnt/boot/grub
```

3. After GRUB is installed on floppy disk #2, the file "stage2" must not be modified, deleted, defragged, or moved. If it is modified in any way, the disk will no longer be bootable. To prevent this, make the file read-only:

```
(DOS)   attrib +r +s stage2

(UNIX)  chmod a-w stage2
```

The DOS command above makes "stage2" a System file as well as Read-only. This is needed to protect against DEFRAG.

NOTE: File "stage1" will be copied into the bootsector. If this file is moved or deleted after GRUB is installed, the disk will still be bootable.

4. Boot your computer from the floppy with GRUB but no filesystem. At the GRUB prompt, eject this floppy and insert the formatted floppy disk (with the filesystem and "stage1" and "stage2" files, possibly in directory "/boot/grub".

5a. If files "stage1" and "stage2" are stored in "/boot/grub" on disk #2, you can install GRUB on disk #2 simply by typing:

```
setup (fd0)
```

This is apparently equivalent to this single command line:

```
install /boot/grub/stage1 d (fd0) /boot/grub/stage2 p
    /boot/grub/menu.lst
```

5b. If files "stage1" and "stage2" are stored elsewhere, e.g. in subdirectory "/foo", install GRUB on the second floppy disk like this (this is also a single command line):

```
        install=(fd0)/foo/stage1 (fd0) (fd0)/foo/stage2 0x8000 p
            (fd0)/foo/menu.lst

Floppy disk #2 (the disk with the filesystem) is now bootable.

xxx - Boot from disk #2, copy new/modified "stage2", and re-run
"setup" or "install"? Will this work? (xxx - GRUB is not a
shell -- it can't copy files, or list directories -- can it?)

xxx - install syntax:
                0x8000
                This value gets embedded into the bootsector
                of the floppy to indicate the address that
                stage2 should be loaded into memory.

                p
                Modifies stage2, to report to the kernel the partition
                that stage 2 was found on (I think).

                (fd0)/boot/grub/menu.lst
                Modifies stage2, and tells it where to load the
                menu.lst (bootmenu) configuration file from.

================================================================
Making a Multiboot kernel
================================================================
Multiboot header
----------------
Whatever its file format, your kernel MUST have a Multiboot
header. This header
1. must be aligned on a dword (4-byte) boundary, and
2. must appear in the first 8K of the kernel file.

*** NOTE: An address within the first 8K of the .text section
is not necessarily within 8K of the start of the file.

ELF kernels
-----------
GRUB understands the ELF file format directly. If your kernel
is ELF, you can use the simple Multiboot header shown here:

    MULTIBOOT_PAGE_ALIGN   equ 1<<0
    MULTIBOOT_MEMORY_INFO  equ 1<<1

    MULTIBOOT_HEADER_MAGIC equ 0x1BADB002
    MULTIBOOT_HEADER_FLAGS equ MULTIBOOT_PAGE_ALIGN | MULTIBOOT_MEMORY_INFO
    CHECKSUM equ -(MULTIBOOT_HEADER_MAGIC + MULTIBOOT_HEADER_FLAGS)

    ; The Multiboot header (in NASM syntax)
        align 4
        dd MULTIBOOT_HEADER_MAGIC
        dd MULTIBOOT_HEADER_FLAGS
        dd CHECKSUM

Put this near the beginning of your kernel startup code, then
build your kernel. After the kernel is built, you can use the
GRUB "mbchk" utility to test if the kernel complies with
Multiboot.

Kernel load address
-------------------
GRUB reads the physical address (load address; LMA) of the
kernel from the ELF file. This value must be
1. at or above 1 meg, and
2. below the end of physical RAM

If the load address is below 1 meg, you get error #7:
        Loading below 1MB is not supported

*** NOTE: This is a limitation of GRUB, not of Multiboot.

If the load address is beyond the end of RAM, you get error #28:
        Selected item cannot fit into memory
```

```
And if you use a very high address like 0xC0000000, the math
apparently overflows, and you get error #7 again.

*** NOTE: "mbchk" does not check for these errors.

Normally, the physical address is the same as the VMA, and is
set either in the linker script or on the linker command line
("ld -Ttext=0x100000 ..."). If your version of 'ld' supports it,
the physical and virtual addresses can be specified separately
in the linker script using 'AT':

        OUTPUT_FORMAT("elf32-i386")
        ENTRY(entry)
        virt = 0xC0000000; /* 3 gig */
        phys = 0x100000; /* 1 meg */
        SECTIONS
        {   .text virt : AT(phys)
            {   code = .;
                *(.text)
                . = ALIGN(4096); }
            .data :  AT(phys + (data - code))
            {   data = .;
                *(.data)
                . = ALIGN(4096); }
            .bss :  AT(phys + (bss - code))
            {   bss = .;
                *(.bss)
                *(COMMON)
                . = ALIGN(4096); }
            end = .; }

After linking, use 'objdump -h' to check that the addresses
are all correct.

DJGPP COFF kernels and other file formats
-----------------------------------------
DJGPP users can make ELF files using these tools:
        http://www.multimania.com/placr/binutils.html

(xxx - this server is often difficult to reach. Someone should
mirror these tools. I'm near my disk quota :)

I recommend building a regular COFF kernel, then doing this:
        objcopy-elf -O elf32-i386 krnl.cof krnl.elf

Failing this, you can make GRUB load a COFF kernel by using the
"aout kludge". This uses additional fields at the end of the
Multiboot header, like this:

    MULTIBOOT_PAGE_ALIGN   equ 1<<0
    MULTIBOOT_MEMORY_INFO  equ 1<<1
    MULTIBOOT_AOUT_KLUDGE  equ 1<<16

    MULTIBOOT_HEADER_MAGIC equ 0x1BADB002
    MULTIBOOT_HEADER_FLAGS equ MULTIBOOT_PAGE_ALIGN | MULTIBOOT_MEMORY_INFO | MULTIBOOT_AOUT_KLUDGE
    CHECKSUM               equ -(MULTIBOOT_HEADER_MAGIC + MULTIBOOT_HEADER_FLAGS)

    ; The Multiboot header
        align 4
    mboot:
        dd MULTIBOOT_HEADER_MAGIC
        dd MULTIBOOT_HEADER_FLAGS
        dd CHECKSUM
    ; fields used if MULTIBOOT_AOUT_KLUDGE is set in MULTIBOOT_HEADER_FLAGS
        dd mboot ; these are PHYSICAL addresses
        dd code  ; start of kernel .text (code) section
        dd edata ; end of kernel .data section
        dd end   ; end of kernel BSS
        dd start ; kernel entry point (initial EIP)

NOTE: The "aout kludge" works with binary and other file
formats, too.   (xxx - untested; should be correct)

================================================================
Booting!
```

```
================================================================
1.  Make sure your kernel is some convenient place where GRUB
    can find it. It need not be on the floppy disk.

2.  Boot from a GRUB floppy.

3.  (Optional) Tell GRUB what device to use for its root
    directory:

        root (hd0,1)

    This "mounts" the 2nd primary partition on the 1st hard
    drive as the root directory.

4.  Tell GRUB where your kernel is:

        kernel /krnl.elf

    If you did not specify the root device, you must give the
    device explicitly at the start of each path name:

        kernel (hd0,1)/krnl.elf

5.  If GRUB has no complaints about the kernel file, boot it:

        boot

================================================================
Passing system information from GRUB to your kernel
================================================================
Upon entry to the 32-bit kernel
1. CS points to a code segment descriptor with base address 0
   and limit 4 gig - 1
2. DS, SS, ES, FS, and GS point to a data segment descriptor
   with base address 0 and limit 4 gig - 1
3. A20 is enabled
4. Paging is disabled
5. Interrupts are disabled
6. EAX=0x2BADB002
7. EBX contains the linear address of (i.e. a pointer to) a
   block of system and bootstrap information:

        /* The Multiboot information.  */
        typedef struct multiboot_info
        {
          unsigned long flags;
          unsigned long mem_lower;
          unsigned long mem_upper;
          unsigned long boot_device;
          unsigned long cmdline;
          unsigned long mods_count;
          unsigned long mods_addr;
          union
          {
            aout_symbol_table_t aout_sym;
            elf_section_header_table_t elf_sec;
          } u;
          unsigned long mmap_length;
          unsigned long mmap_addr;
        } multiboot_info_t;

This information can be accessed from C code by pushing the
pointer in EBX onto the stack before calling main():

asm startup code:
        ...
        push ebx
        call _main ; "call main" for Linux/ELF
        ...

C code:
        #include
        ...
        int main(multiboot_info_t *boot_info)
        {       if(boot_info->flags & 2)
```

```
                {       kprintf("the command line is:\n'%s'\n",
                                (char *)boot_info->cmdline); }
                ...
```

xxx - more info

```
================================================================
Making a boot menu (file "menu.lst")
================================================================
Example 1:
        # Entry 0:
        title   WildMagnolia
        root    (fd0)
        kernel  /boot/kernel.elf
        module  /boot/mod_a
        module  /boot/mod_b

Example 2:
        #
        # Sample boot menu configuration file
        #

        #  default - boot the first entry.
        default 0

        # if have problem - boot the second entry.
        fallback 1

        # after 30 sec boot default.
        timeout 30

        # GNU Hurd
        title  GNU/Hurd
        root   (hd0,0)
        kernel /boot/gnumach.gz root=hd0s1
        module /boot/serverboot.gz

        # Linux - boot ot second HDD
        title  GNU/Linux
        kernel (hd1,0)/vmlinuz root=/dev/hdb1

        # booting Mach - get kernel from floppy
        title  Utah Mach4 multiboot
        root   (hd0,2)
        pause  Insert the diskette now□!!
        kernel (fd0)/boot/kernel root=hd0s3
        module (fd0)/boot/bootstrap

        # booting OS/2
        title OS/2
        root  (hd0,1)
        makeactive
        # chainload OS/2 bootloader from the first sector
        chainloader +1

        # For booting Windows NT or Windows95
        title Windows NT / Windows 95 boot menu
        root        (hd0,0)
        makeactive
        chainloader +1
        # za boot na DOS ako Windows NT e instaliran
        # chainload /bootsect.dos

        # Colors change :0).
        title Change the colors
        color light-green/brown blink-red/blue

================================================================
Loading modules with the kernel
================================================================
xxx

================================================================
Other
================================================================
```

- gzip-compressed kernels?
- does GRUB understand kernel file formats other than ELF?

by K.J.

Yes, you may also compile your kernel as a COFF file or an AOUT file.