

Pascal Bare Bones

From OSDev Wiki

Credit flies to De Deyn Kim for the freepascal, public domain version of BareBones.

Tools needed to build the project:

- FPC
- NASM
- binutils(ld) built with elf support

Difficulty level



Medium

Contents

- 1 stub.asm
- 2 kernel.pas
- 3 console.pas
- 4 multiboot.pas
- 5 system.pas
- 6 Linker script
- 7 Compiling and Linking the modules

stub.asm

```
; //////////////////////////////////////
; //                                     //
; //           Freepascal barebone OS   //
; //           stub.asm                  //
; //                                     //
; //////////////////////////////////////
; //
; //      By:           De Deyn Kim <kimdedeyn@skynet.be>
; //      License:      Public domain
; //

;
; Kernel stub
;

;
; We are in 32bits protected mode
;
[bits 32]
```

```

;
; Export entrypoint
;
[global kstart]

;
; Import kernel entrypoint
;
[extern kmain]

;
; Possible multiboot header flags
;
MULTIBOOT_MODULE_ALIGN          equ      1<<0
MULTIBOOT_MEMORY_MAP            equ      1<<1
MULTIBOOT_GRAPHICS_FIELDS       equ      1<<2
MULTIBOOT_ADDRESS_FIELDS        equ      1<<16

;
; Multiboot header defines
;
MULTIBOOT_HEADER_MAGIC          equ      0x1BADB002
MULTIBOOT_HEADER_FLAGS          equ      MULTIBOOT_MODULE_ALIGN |
MULTIBOOT_HEADER_CHECKSUM       equ      -(MULTIBOOT_HEADER_MAGIC

;
; Kernel stack size
;
KERNEL_STACKSIZE                equ      0x4000

section .text

;
; Multiboot header
;
align 4
dd MULTIBOOT_HEADER_MAGIC
dd MULTIBOOT_HEADER_FLAGS
dd MULTIBOOT_HEADER_CHECKSUM

;
; Entrypoint
;
kstart:
    mov esp, KERNEL_STACK+KERNEL_STACKSIZE ;Create kernel s
    push eax                               ;Multiboot magic
    push ebx                               ;Multiboot info
    call kmain                             ;Call kernel ent
    cli                                    ;Clear interrupt
    hlt                                    ;Halt machine

```

```

section .bss

;
; Kernel stack location
;
align 32
KERNEL_STACK:
    resb KERNEL_STACKSIZE

```



kernel.pas

```

{
//////////////////////////////////////
//                                                                    //
//          Freepascal barebone OS          //
//          kernel.pas                      //
//                                                                    //
//////////////////////////////////////
//
//      By:          De Deyn Kim <kimdedeyn@skynet.be>
//      License:     Public domain
//
}

unit kernel;

interface

uses
    multiboot,
    console;

procedure kmain(mbinfo: Pmultiboot_info_t; mbmagic: DWORD); stdcall;

implementation

procedure kmain(mbinfo: Pmultiboot_info_t; mbmagic: DWORD); stdcall
begin
    kclearscreen();
    kwritestr('Freepascal barebone OS booted!');
    xpos := 0;
    ypos += 1;

    if (mbmagic <> MULTIBOOT_BOOTLOADER_MAGIC) then
    begin
        kwritestr('Halting system, a multiboot-compliant
asm

```

```

                                cli
                                hlt
                                end;
end
else
begin
    kwritestr('Booted by a multiboot-compliant boot
xpos := 0;
ypos += 2;
kwritestr('Multiboot information:');
xpos := 0;
ypos += 2;
kwritestr('                                Lower memory
kwriteint(mbinfo^.mem_lower);
kwritestr('KB');
xpos := 0;
ypos += 1;
kwritestr('                                Higher memory
kwriteint(mbinfo^.mem_upper);
kwritestr('KB');
xpos := 0;
ypos += 1;
kwritestr('                                Total memory
kwriteint(((mbinfo^.mem_upper + 1000) div 1024)
kwritestr('MB');
                                end;

                                asm
                                @loop:
                                jmp @loop
                                end;
end;

end.

```

console.pas

```

{
////////////////////////////////////
//                                                                    //
//          Freepascal barebone OS          //
//          console.pas                      //
//                                                                    //
////////////////////////////////////
//
//          By:          De Deyn Kim <kimdedeyn@skynet.be>
//          License:     Public domain
//

```

```

}

unit console;

interface

var
    xpos: Integer = 0;
    ypos: Integer = 0;

procedure kclearscreen();
procedure kwritechr(c: Char);
procedure kwritestr(s: PChar);
procedure kwriteint(i: Integer);
procedure kwritedword(i: DWORD);

implementation

var
    vidmem: PChar = PChar($b8000);

procedure kclearscreen(); [public, alias: 'kclearscreen'];
var
    i: Integer;
begin
    for i := 0 to 3999 do
        vidmem[i] := #0;
    end;

procedure kwritechr(c: Char); [public, alias: 'kwritechr'];
var
    offset: Integer;
begin
    if (ypos > 24) then
        ypos := 0;

    if (xpos > 79) then
        xpos := 0;

    offset := (xpos shl 1) + (ypos * 160);
    vidmem[offset] := c;
    offset += 1;
    vidmem[offset] := #7;
    offset += 1;

    xpos := (offset mod 160);
    ypos := (offset - xpos) div 160;
    xpos := xpos shr 1;

end;

```

```

procedure kwritestr(s: PChar); [public, alias: 'kwritestr'];
var
    offset, i: Integer;
begin
    if (ypos > 24) then
        ypos := 0;

    if (xpos > 79) then
        xpos := 0;

    offset := (xpos shl 1) + (ypos * 160);
    i := 0;

    while (s[i] <> Char($0)) do
        begin
            vidmem[offset] := s[i];
            offset += 1;
            vidmem[offset] := #7;
            offset += 1;
            i += 1;
        end;

        xpos := (offset mod 160);
        ypos := (offset - xpos) div 160;
        xpos := xpos shr 1;
    end;

procedure kwriteint(i: Integer); [public, alias: 'kwriteint'];
var
    buffer: array [0..11] of Char;
    str: PChar;
    digit: DWORD;
    minus: Boolean;
begin
    str := @buffer[11];
    str^ := #0;

    if (i < 0) then
        begin
            digit := -i;
            minus := True;
        end
    else
        begin
            digit := i;
            minus := False;
        end;

    repeat
        Dec(str);

```

```

        str^ := Char((digit mod 10) + Byte('0'));
        digit := digit div 10;
    until (digit = 0);

    if (minus) then
    begin
        Dec(str);
        str^ := '-';
    end;

    kwritestr(str);
end;

procedure kwritedword(i: DWORD); [public, alias: 'kwritedword'];
var
    buffer: array [0..11] of Char;
    str: PChar;
    digit: DWORD;
begin
    for digit := 0 to 10 do
        buffer[digit] := '0';

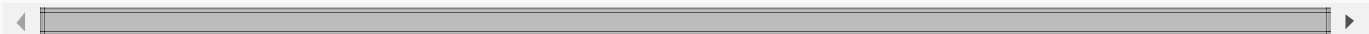
    str := @buffer[11];
    str^ := #0;

    digit := i;
    repeat
        Dec(str);
        str^ := Char((digit mod 10) + Byte('0'));
        digit := digit div 10;
    until (digit = 0);

    kwritestr(@Buffer[0]);
end;

end.

```



multiboot.pas

```

unit multiboot;

interface

const
    KERNEL_STACKSIZE = $4000;

    MULTIBOOT_BOOTLOADER_MAGIC = $2BADB002;

```

type

```
Pelf_section_header_table_t = ^elf_section_header_table_t
elf_section_header_table_t = packed record
    num: DWORD;
    size: DWORD;
    addr: DWORD;
    shndx: DWORD;
end;

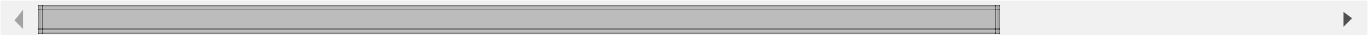
Pmultiboot_info_t = ^multiboot_info_t;
multiboot_info_t = packed record
    flags: DWORD;
    mem_lower: DWORD; { Amount of memory available below 1
    mem_upper: DWORD; { Amount of memory available above 1
    boot_device: DWORD;
    cmdline: DWORD;
    mods_count: DWORD;
    mods_addr: DWORD;
    elf_sec: elf_section_header_table_t;
    mmap_length: DWORD;
    mmap_addr: DWORD;
end;

Pmodule_t = ^module_t;
module_t = packed record
    mod_start: DWORD;
    mod_end: DWORD;
    name: DWORD;
    reserved: DWORD;
end;

Pmemory_map_t = ^memory_map_t;
memory_map_t = packed record
    size: DWORD;
    { You can declare these two as a single qword if your
    base_addr_low: DWORD;
    base_addr_high: DWORD;
    { And again, these can be made into one qword variable
    length_low: DWORD;
    length_high: DWORD;
    mtype: DWORD;
end;
```

implementation

end.



system.pas

```
unit system;

interface

type
  cardinal = 0..$FFFFFFFF;
  hresult = cardinal;
  dword = cardinal;
  integer = longint;

  pchar = ^char;

implementation

end.
```

Linker script

linker.script

```
ENTRY(kstart)
SECTIONS
{
  .text 0x100000 :
  {
    text = .; _text = .; __text = .;
    *(.text)
    . = ALIGN(4096);
  }
  .data :
  {
    data = .; _data = .; __data = .;
    *(.data)
    kimage_text = .;
    LONG(text);
    kimage_data = .;
    LONG(data);
    kimage_bss = .;
    LONG(bss);
    kimage_end = .;
    LONG(end);
    . = ALIGN(4096);
  }
  .bss :
  {
    bss = .; _bss = .; __bss = .;
    *(.bss)
    . = ALIGN(4096);
  }
  end = .; _end = .; __end = .;
}
```

Compiling and Linking the modules

Assemble stub.asm with:

```
nasm -f elf stub.asm -o stub.o
```

The Pascal modules with:

```
fpc -Aelf -n -O3 -Op3 -Si -Sc -Sg -Xd -CX -XXs -Rintel -Tlinux kernel.pas
```

- -Aelf - instructs the internal fpc assembler to output an ELF object.;
- -n - ignores fpc.cfg;
- -O3 - perform level 3 optimizations;
- -Op3 - tune code for PentiumPro / P-II / Cyrix 6x86 / K6 (TM);
- -Si - enable C++ style INLINE keyword;
- -Sc - enable C style operators;
- -Sg - enable support for labels;
- -Xd - tells the compiler to forget the standard library path;
- -CX - tells the compiler to create smartlinkable units
- -XXs - tells the compiler to do smartlinking (-XX) and debugging symbols stripping (-Xs)
- -Rintel - sets the inline assembly syntax to intel style;
- -Tlinux - specifies that the target operating system is Linux. (Don't even ask!)

Then link the whole thing with:

```
ld --gc-sections -s -Tlinker.script -o kernel.obj stub.o kernel.o multiboot.o system.o console.o
```

- --gc-sections -s, in combination with -CX -XXs above, eliminates RTTI symbols from resulting binary

Retrieved from "http://wiki.osdev.org/index.php?title=Pascal_Bare_Bones&oldid=12464"

Categories: Level 2 Tutorials | Bare bones tutorials

-
- This page was last modified on 28 January 2012, at 19:20.
 - This page has been accessed 17,870 times.