

# Jadi Anda Ingin Membuat Sendiri Sistem Operasi x86?

By [Patrick Mahoney](#)

## 1. Perkenalan

Salah satu kesulitan besar yang dihadapi oleh programmer hobbyist ketika mulai mencoba mengembangkan sistem operasinya sendiri adalah menentukan dari mana ia harus memulai. Banyak buku menjelaskan dengan mendalam konsep sistem operasi secara teoritis, namun tidak satu pun yang tampaknya bisa membawa programmer hobbyist untuk memahami konsep tersebut. Ini adalah apa yang akan dilakukan oleh artikel ini.

Beberapa artikel yang berhubungan dengan topik ini muncul di beberapa edisi terakhir Linux Gazette. Saya merencanakan untuk melakukan pendekatan dengan menggunakan sesedikit mungkin gaya yang berorientasi pemrograman, dan hanya akan menunjukkan kepada pembaca tool dan tips yang akan ia butuhkan untuk memulai pengembangan dari sistem operasinya. Sekali ia membaca artikel ini, pembaca yang tertarik seharusnya segera mulai browsing segala sesuatu yang ia perlukan yang tersedia dan memulai untuk mendesain dan mengetikkan kode program.

Anda mungkin tidak mengerti, bahwa pengembangan suatu sistem operasi tidak dimulai dari awal. (!) Menulis sebuah bootloader yang bagus akan menjadi keseluruhan proyek itu sendiri, dan saya tidak menyarankan anda untuk memulai sebuah proyek pengembangan sistem operasi dengan menulis sebuah bootloader. Banyak bootloader yang handal tersedia dengan bebas (Grub, lilo, ppcboot, dan lain-lain...). Jika anda berencana untuk menulisnya sendiri, saya menyarankan untuk menunda pekerjaan ini pada bagian lain dari proyek. Pada artikel ini, saya akan menggunakan GNU Grub, Grand Unified Bootloader.

## 2. Penjelasan mengenai lingkungan pengembangan

Untuk mengurangi kesulitan pengembangan sistem operasi, anda harus melakukan pengaturan sebuah lingkungan pengembangan yang disesuaikan, yang memenuhi beberapa syarat sebagai berikut:

- Anda harus segera mengetes kernel yang baru saja dicompile
- Anda tidak boleh mereboot mesin yang anda gunakan untuk pengembangan
- Anda tidak boleh menggunakan floppy sebagai media penyimpanan untuk sistem operasi anda

Artikel ini akan menghadirkan satu dari beberapa lingkungan yang mungkin yang sesuai dengan syarat di atas, yang terdiri dari satu mesin pengembangan dan sebuah mesin pengujian, yang keduanya berada pada jaringan yang umum.

### 2.1 Mesin pengembangan

Tak dapat dihindari, mesin ini harus dilengkapi dengan satu set tool pemrograman yang baik: compiler assembly dan C, sebuah linker dan sebuah utilitas 'make' adalah suatu keharusan.

Sebuah tool yang lebih berguna daripada yang saya pikirkan adalah sebuah emulator. Tool ini akan membantu anda melakukan debug pada kernel dan akan mengijinkan anda

untuk segera mengetes baris kode yang baru saja anda tambahkan. Namun jangan terlalu mudah dibodohi, sebuah emulator tidak akan bisa menggantikan sebuah mesin pengujian yang bagus.

Selanjutnya, anda membutuhkan sebuah server TFTP. Tool ini akan memungkinkan bootloader mesin pengujian yang tftp-enabled untuk mengambil kernel dari mesin pengembangan melalui koneksi jaringan.

## 2.2 Mesin pengujian

Yang dibutuhkan mesin ini adalah sebuah network card dan sebuah bootloader yang tftp-enabled yang mendukungnya.

## 3. Mengatur lingkungan pengembangan

### 3.1 Mesin pengembangan

Tool programming yang dipilih adalah:

- gcc 2.95.4
- ld 2.13.90.0.10

Bochs versi 1.4.1 adalah emulator x86 yang dipilih. Perlakuan khusus harus dilakukan untuk mengcompilanya dengan debugger mode enabled. Perintah yang digunakan adalah:

```
$ ./configure --enable-x86-debugger  
$ make
```

Agar Bochs bisa digunakan dengan baik, anda harus membuat sebuah disk image. Image ini harus memiliki sebuah bootloader dan sebuah filesystem. Ini bisa dilakukan dengan menggunakan script [mkbimage](#). Jika anda malas untuk melakukannya sendiri, ambillah [gzipped 10MB disk image](#) ini dan tambahkan

```
diskc: file=c.img, cyl=24, heads=16, spt=63
```

pada file .bochrc.

Sebagai TFTP server, saya menggunakan atftpd. Ini adalah implementasi linux-based TFTP server yang mudah digunakan.

### 3.2 Mesin Pengujian

Bootloader yang dipilih adalah GNU Grub versi 0.92. Perlakuan khusus harus dilakukan untuk mengenable tftp client pada Grub untuk bisa berkomunikasi dengan network card. Mesin pengujian yang saya gunakan menggunakan NE2000 ISA clone yang murah. Mengikuti instruksi netboot/README.netboot dengan hati-hati, saya menggunakan perintah ini:

```
$ ./configure --enable-ne --enable-ne-scan=0x220  
$ make
```

Ingat bahwa PnP PCI card akan lebih mudah untuk dikonfigurasi. Sekarang, anda dapat menginstal image Grub pada MBR mesin pengujian atau pada floppy di mana mesin pengujian anda akan diboot. Saya lebih memilih yang terakhir, karena mesin pengujian juga digunakan untuk keperluan lain, dan oleh karena itu, saya memilih untuk tidak bermain-main dengan harddisknya.

```
$ cat ./stage1/stage1 ./stage2/stage2 > /dev/fd0
```

Sekarang tinggal memasukkan floppy pada mesin pengujian untuk melihat apakah network card anda bisa dikenali. Anda bisa mengkonfigurasinya sendiri atau menggunakan dhcp server, jika ada.

```
grub> dhcp
Probing... [NE*000]
NE2000 base 0x220, addr 00:C0:A8:4E:5A:76
Address: 192.168.22.14
Netmask: 255.255.255.0
Server: 192.168.22.1
Gateway: 192.168.22.1
```

Ingat bahwa anda tidak perlu mengkonfigurasi parameter-parameter tersebut secara manual tiap kali anda melakukan boot. Lihatlah dokumentasi GNU Grub dan script 'grub-install' untuk lebih jelasnya.

That's it! Anda siap untuk melakukan tes!

#### 4. Mengetes pengaturan lingkungan pengembangan anda...

Seperti yang telah saya sebutkan, saya akan membiarkan inti pemrograman sistem operasi kepada para ahlinya. Sehingga untuk mengetes pengaturannya, kita akan menggunakan contoh kernel dari source GNU Grub yang berada di direktori /docs.

Kernel tersebut dibangun dari tiga file source: boot.S, kernel.c dan multiboot.h. Anda bisa membangun kernel tersebut dengan:

```
$ gcc -I. -c ./boot.S
$ gcc -I. -c ./kernel.c
$ ld ./kernel.o ./boot.o -o kernel -Ttext 100000
```

Berikut adalah penjelasan singkat dan tidak lengkap. Multiboot adalah sebuah standar yang mendefinisikan sebuah cara bagi bootloader untuk melewatkan informasi menuju kernel yang akan dimuat. boot.S menerima informasi tersebut, mengeset sebuah stack (tumpukan), dan memanggil 'cmain'. Fungsi ini akan mengeset vga display, membaca informasi yang dilewatkan kepadanya, menampilkan beberapa pesan dan kemudian pergi. Lalu, boot.S kembali memegang kendali, menampilkan string 'Halted', dan memasuki loop yang tidak terbatas. Sangat sederhana, bukan? Pembaca dipersilakan untuk menggali kodenya untuk lebih detailnya.

#### 4.1 ...dengan Bochs

Rencananya adalah memount disk image anda melewati sebuah loopback device, menyalin kernel anda pada filesystem image tersebut, meng-unmount image, dan menjalankan Bochs. Tentunya, anda harus menambahkan sebuah offset untuk menjalankan filesystem. Tetapi anda sudah tahu, kan?

```
# /sbin/losetup -o 32256 /dev/loop1 ./c.img
# /bin/mount -t ext2 /dev/loop1 /mnt/osdev/
# cp /docs/kernel /mnt/osdev
# umount /mnt/osdev/
# /sbin/losetup /dev/loop1 -d
$ bochs
```

Tentunya, hal di atas dapat diotomatisasi dengan Makefile. Pada Grub, lakukan:

```
grub> kernel (hd0,0)/kernel
grub> boot
```



(Klik pada gambar untuk ukuran penuh.)

## 4.2 ...dengan mesin pengujian anda

Pertama, atur TFTP server anda sehingga client dapat mengambil kernel anda:

```
# /usr/sbin/atftpd --daemon /home/bono/src/grub-0.92/docs
```

Jalankan mesin pengujian. Konfigurasi koneksi jaringan anda seperti ditunjukkan di atas. Selanjutnya, tentukan alamat IP mesin pengembangan anda seperti alamat TFTP server dan lokasi dari kernel image. Ingat bahwa pilihan ini dapat diset oleh dhcp server. Akhirnya, mulailah proses boot.

(...)

```
grub> tftpsrv 192.168.22.36
Address: 192.168.22.14
Netmask: 255.255.255.0
Server: 192.168.22.36
Gateway: 192.168.22.1
```

```
grub> kernel (nd)/kernel
[Multiboot-elf, <0x100000:0x807:0x0>, <0x101808:0x0:0x4018>,
shtab=0x106190, entry=0x100568]
```

```
grub> boot
```

Sebuah tampilan yang mirip dengan Bochs di atas akan muncul pada layar mesin pengujian.

## 5. Kemana lagi setelah ini

Anda sudah begitu siap untuk memulai pengembangan sistem operasi. Banyak sekali dokumentasi yang bagus yang ada di web. Browse, kirim, tanya, dan berpikirlah. Monolithic atau micro kernel? Segmentation atau paging?

Jika kebutuhan debugging anda menjadi lebih besar daripada emulator dan kernel, sebuah pengaturan yang bisa anda tambahkan pada sistem operasi adalah serial debugger. Ini bisa bermacam-macam, dari beberapa bytes yang dilemparkan ke serial port, sampai sebuah gdb-compatible remote-debugging extension. Informasi ini bisa didapatkan dan diproses oleh mesin pengembangan anda melewati sebuah null-modem serial cable. Ini adalah suatu kebiasaan yang berguna dalam pengembangan sistem operasi.

## 6. Resources

- [Tanenbaum' os dev book](#)

Buku pegangan bagi pengembangan sistem operasi

- [alt.os.development](#)

Disana anda akan mendapatkan solusi untuk permasalahan anda!

- Freenode IRC's #osdev (irc.debian.org)

Orang-orang yang bersahabat yang tidak pernah tidur!

- [Beberapa tutorial pengembangan sistem operasi](#) termasuk dari Tim Robinson.

Tim pernah berada di sana!

- [Pusat Resource Sistem Operasi](#)
- [BosoKernel](#)

Tutorial pemula x86 yang manis. (Perancis)

- [Intel Architecture Software Developer's Manual Volume 3: System Programming](#)

Jangan meninggalkan rumah tanpanya.

### **1.1. 7. Terimakasih**

Terimakasih banyak kepada semua orang yang telah menerima dengan sabar untuk menjawab pertanyaan-pertanyaan saya yang tidak pernah ada habisnya di #osdev: pavloskii, geist, oink, byrdkernel, air.

---

Copyright © 2002, Patrick Mahoney. Copying license <http://www.linuxgazette.com/copying.html>  
Published in Issue 85 of *Linux Gazette*, December 2002

---

Diterjemahkan oleh [Triyan W. Nugroho](#)