

# **Tugas Besar Milestone 2**

IF2230 - Sistem Operasi

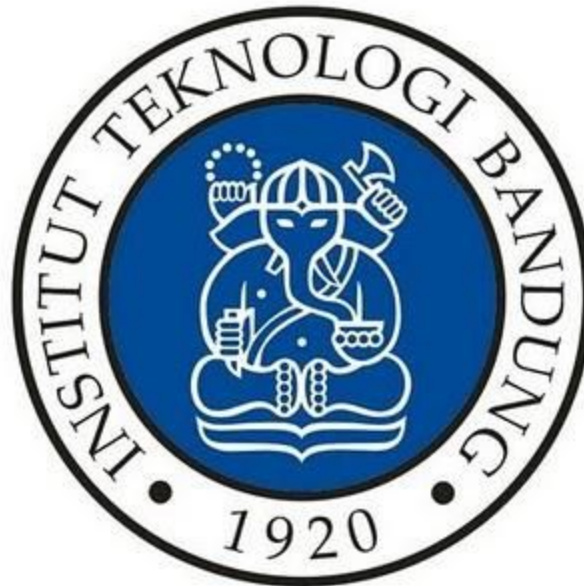
Diajukan sebagai tugas dari mata kuliah Sistem Operasi di jurusan Teknik Informatika Institut Teknologi Bandung

Oleh :

**Farhan Amin (13515043)**

**Dery Rahman A (13515097)**

**Aulia Ichsan Rifkyano (13515100)**



**PROGRAM STUDI INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
2017**

## A. Jawaban Pertanyaan

1. Bagaimana program dieksekusi di dalam OS?

### Jawaban:

Program dijalankan dengan 2 tahap, yakni *load* dan *execute*. Langkah-langkah yang dilakukan adalah sebagai berikut:

1. Program di-*load* dari *disk* lalu disimpan di sebuah *buffer*.
2. Program yang sudah terdapat di *buffer* dipindahkan ke dasar *memory segment* (dengan kelipatan 0x1000) yang sesuai. Ada beberapa *segment* yang tidak boleh digunakan seperti *segment* 0x000 yang digunakan sebagai tempat menyimpan *interrupt vectors*, *segment* 0x1000 yang digunakan sebagai tempat penyimpanan kernel yang dijalankan oleh *bootloader*, *segment- segment* yang lebih dari 0xA000 yang sudah dipesan yang nantinya akan digunakan sebagai tempat video RAM, *memory*, *I/O Devices*, dan BIOS.
3. *Segment registers* diatur agar mengandung program yang ingin dijalankan, lalu atur *stack pointer* menjadi *stack* dari program.
4. *Jump* ke awal *segment* yang digunakan program untuk menjalankan program tersebut.

Didalam OS, pada sector 0, diisi oleh *bootloader*, sector 1 diisi oleh *map.img* dan sector 2 diisi oleh *dir.img*. Sector 3 dan seterusnya diisi oleh program yang nantinya akan dieksekusi. *Map.img* digunakan untuk menandai sector mana saja yang telah dipakai. Dalam hal ini tentu saja sector 0, 1, dan 2 bernilai true (0xFF), karena telah digunakan untuk *bootloader*, *map*, dan *directory*.

Program di-*load* kemudian diisi ke dalam memory seperti yang telah dijelaskan diatas.

2. Apa yang membedakan *user program* dengan *kernel program* di OS buatan kalian?

### Jawaban:

*Kernel program* dijalankan pada *kernel space*, sedangkan *user program* dijalankan pada *user space*. Kernel memiliki *super access mode* (bisa mengakses apapun yang ada pada sistem). Sedangkan pada *user program* tidak. *User program* yang ingin melakukan hal-hal yang tidak bisa dilakukan di *user space* harus melakukan *interrupt* melalui *system call* supaya bisa memasuki *kernel space*.

3. Apa yang terjadi jika Anda melakukan *execute* untuk file yang bukan program? Bagaimana cara OS modern mengatasi hal tersebut?

### Jawaban:

Saat melakukan eksekusi, OS akan menginterpretasikan tiap *bytes* dari file sebagai instruksi dalam bahasa mesin. Jika file yang di eksekusi bukan merupakan *executable* file, maka akan terjadi *undefined behaviour*.

Yang dilakukan OS modern untuk mengatasi hal tersebut adalah dengan mengharuskan *executable file* diawali dengan *magic number*. Sebagai contoh, semua program Java yang telah

di-compile diawali dengan hex number `0xCAFEBAFE`. Compiler dari program yang bersangkutan selalu membuat *magic number* yang sesuai dengan *file* yang dihasilkannya.

4. Bagaimana kernel menghindari copy memory dari kernel space ke user space dan sebaliknya?

**Jawaban:**

Proses penghindaran dilakukan dengan metode *zero copy* yang dapat dilakukan dengan cara mapping file.

5. Apa yang perlu dilakukan untuk meningkatkan file size maximum?

**Jawaban:**

Dengan menambahkan jumlah *bytes per entry* pada *disk*. Penambahan ini menyebabkan kita bisa menyimpan lebih banyak *sector* dalam satu *entry / file*. Efeknya adalah satu file dapat mengacu ke lebih banyak *sector*. Jumlah *bytes per entry* pada awalnya adalah 32 byte, dengan meningkatkan jumlah *byte* maka file size maximum dapat ditingkatkan.

Efeknya adalah *sector* yang dibutuhkan untuk *file* yang besar membutuhkan jumlah *sector* yang banyak. Oleh karena itu, perlu ditingkatkan jumlah *sector*. Kami mencoba untuk meningkatkan jumlah *sector* yang dipakai, yang tadinya 10, menjadi 20 untuk mengukung karena banyaknya file yang dibuat pada `p4_extended_kernel`.

## BONUS

Sebuah entry memiliki 32 bytes di mana 6 bytes digunakan untuk nama file dan 6 bytes digunakan untuk nama parent directory, sisa 20 bytes nya untuk menentukan apakah dia sebuah file (dengan bytesnya yang terisi sectors) atau sebuah directory di mana semua bytes sisanya berisi 00. Untuk memakan sedikit space saja, bisa digunakan byte ke 13 saja untuk menentukan apakah dia sebuah file atau directory. Apabila directory, maka hanya memakan 13 bytes, namun jika folder tetap memakan 32 bytes. Tentu hal ini akan mengurangi jumlah *space* yang digunakan, dan sangat berpengaruh jika jumlah directory sangat banyak.

Iterasi di `writefile` bisa diubah yang awalnya

```
for (i = 0; i < 6; i++)
    dir[directoryLine*32+i] <- filename[i];
for (i = 6; i < 12; i++)
    dir[directoryLine*32+i] <- wdir[i-6];
```

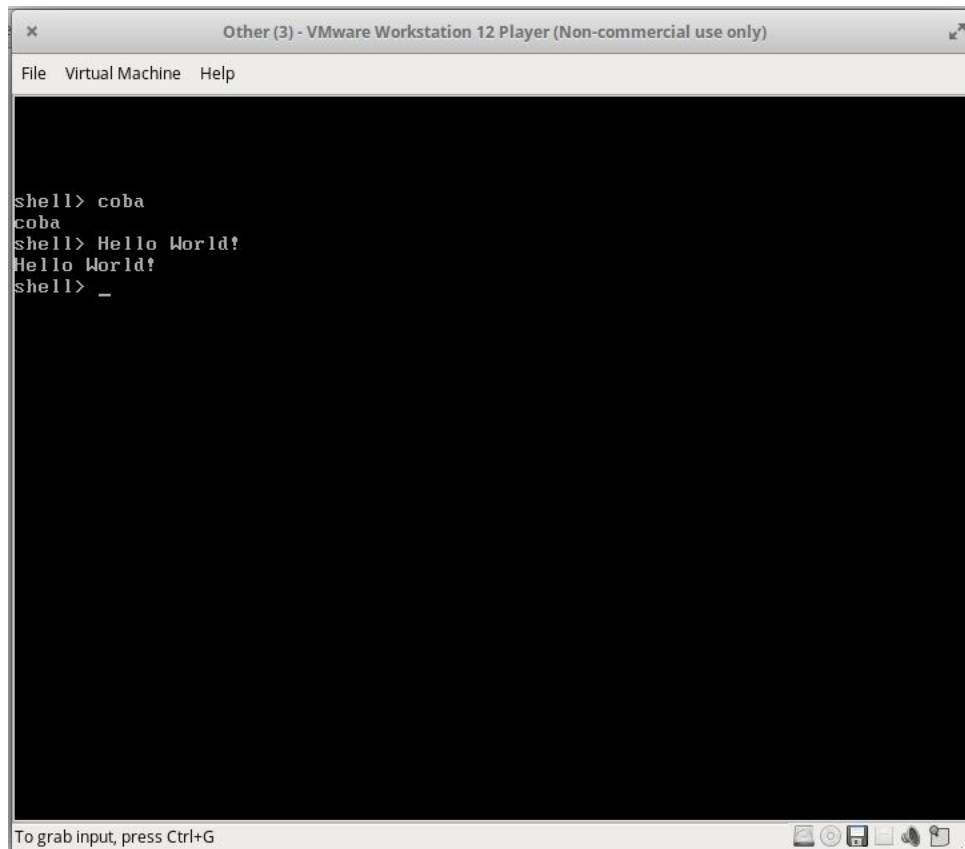
```
menjadi
If (createFile) then
    for (i=0; i<6; i++) do
        dir[directoryLine*32+i] <- filename[i]
Else //if create directory
    for (i=0; i<6; i++) do
        dir[directoryLine*13+i] <- filename[i]
```

## **B. Kesulitan Saat Pengerjaan**

Kesulitan terdapat pada koding saat mengubah OS menjadi *tree structured file system* di mana setiap file akan memiliki folder parent tersendiri dengan batas 6 byte. Kami mencoba mengubah pada kernelnya. Ada berbagai macam fungsi pada kernel di `p4_extended_kernel` yang kami ubah. Kemudian aja juga fungsi-fungsi tambahan untuk mendukung keperluan tersebut.

Sampai saat ini, kami berhasil membuat beberapa fungsi yang setara seperti OS pada umumnya. Antara lain, `createdir`, `cd`, `create` (untuk membuat file), `pwd` (untuk melihat path pada current directory), dan `delete`

## **C. Screenshot Hasil Program**



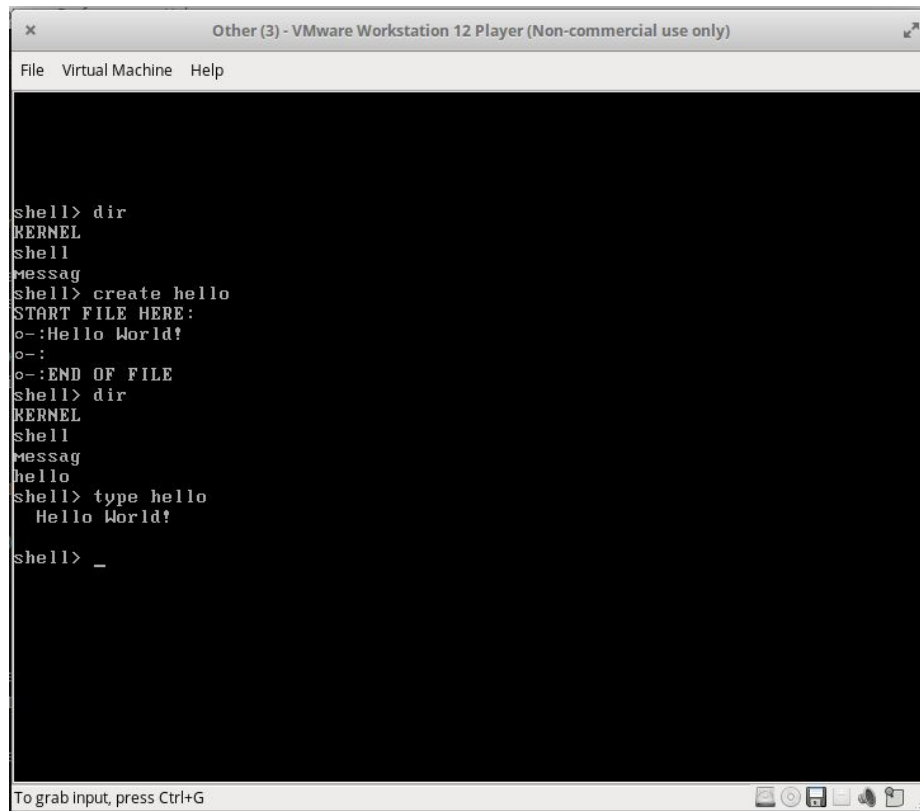
Gambar 1. P2\_kernel : Membaca string kemudian print string tersebut pada Shell

```
Other (3) - VMware Workstation 12 Player (Non-commercial use only)
File Virtual Machine Help

shell> dir
KERNEL
shell
messag
uprog1
shell> type messag
If this message prints out, then your readSector function is working correctly!
shell> execute uprog1
Hello, world!
shell> _

To grab input, press Ctrl+G
```

Gambar 2. P3\_kernel : menuliskan isi dari messag menggunakan type dan eksekusi program uprog1

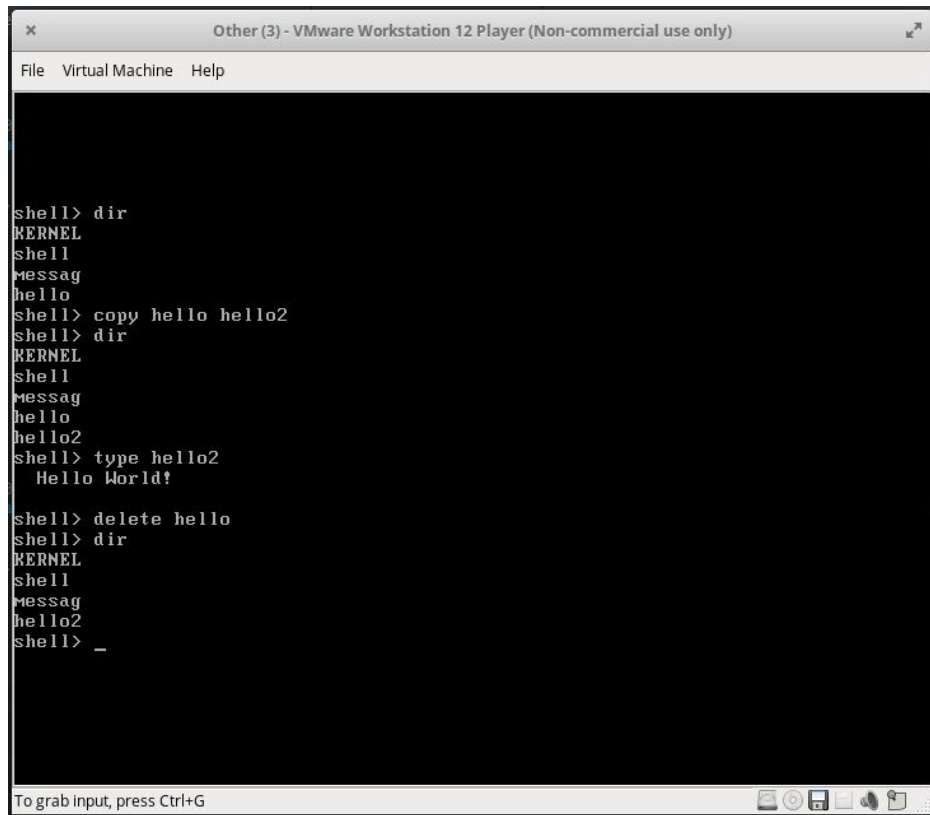


```
Other (3) - VMware Workstation 12 Player (Non-commercial use only)
File Virtual Machine Help

shell> dir
KERNEL
shell
messag
shell> create hello
START FILE HERE:
o-:Hello World!
o-:
o-:END OF FILE
shell> dir
KERNEL
shell
messag
hello
shell> type hello
Hello World!
shell> _

To grab input, press Ctrl+G
```

Gambar 3. P4\_kernel : menjalankan perintah dir dan create pada shell



The image shows a VMware Workstation 12 Player window titled "Other (3) - VMware Workstation 12 Player (Non-commercial use only)". The window contains a terminal window with a black background and white text. The terminal session shows the following commands and outputs:

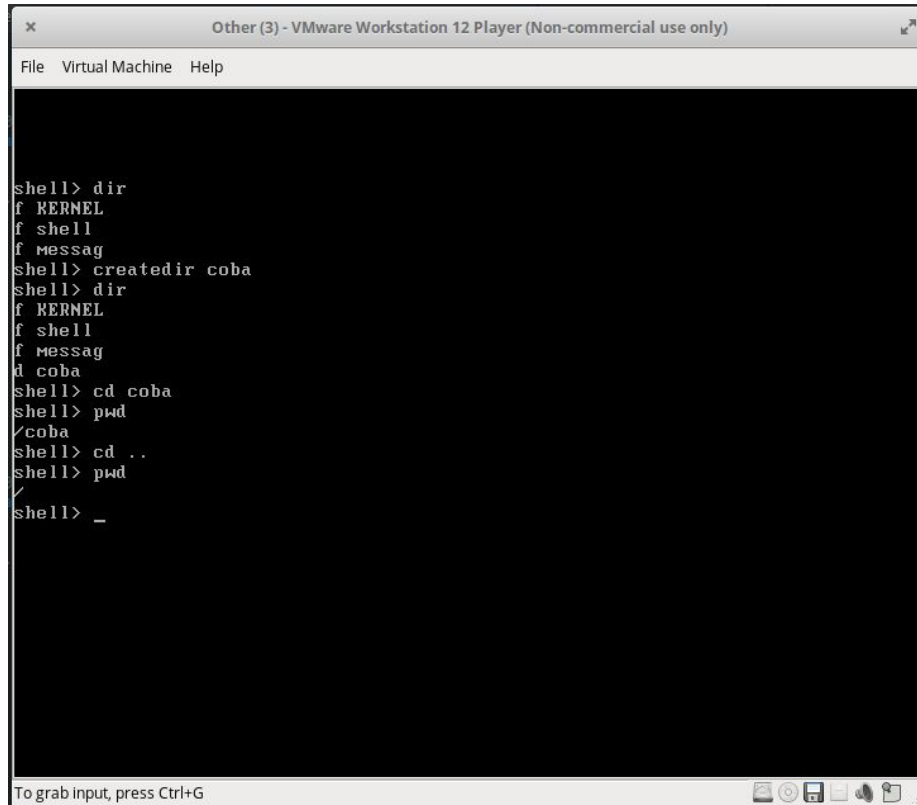
```
shell> dir
KERNEL
shell
messag
hello
shell> copy hello hello2
shell> dir
KERNEL
shell
messag
hello
hello2
shell> type hello2
Hello World!

shell> delete hello
shell> dir
KERNEL
shell
messag
hello2
shell> _
```

At the bottom of the terminal window, there is a status bar that says "To grab input, press Ctrl+G" and several icons.

Gambar 4. P4\_kernel : menjalankan perintah copy dan delete pada shell





```
shell> dir
f KERNEL
f shell
f messag
shell> createdir coba
shell> dir
f KERNEL
f shell
f messag
d coba
shell> cd coba
shell> pwd
/coba
shell> cd ..
shell> pwd
/
shell> _
```

Gambar 5. P4\_extended\_kernel : membuat direktori baru dengan createdir, mengganti posisi direktori dengan fungsi cd (change directory), dan melihat lokasi direktori saat ini dengan fungsi pwd

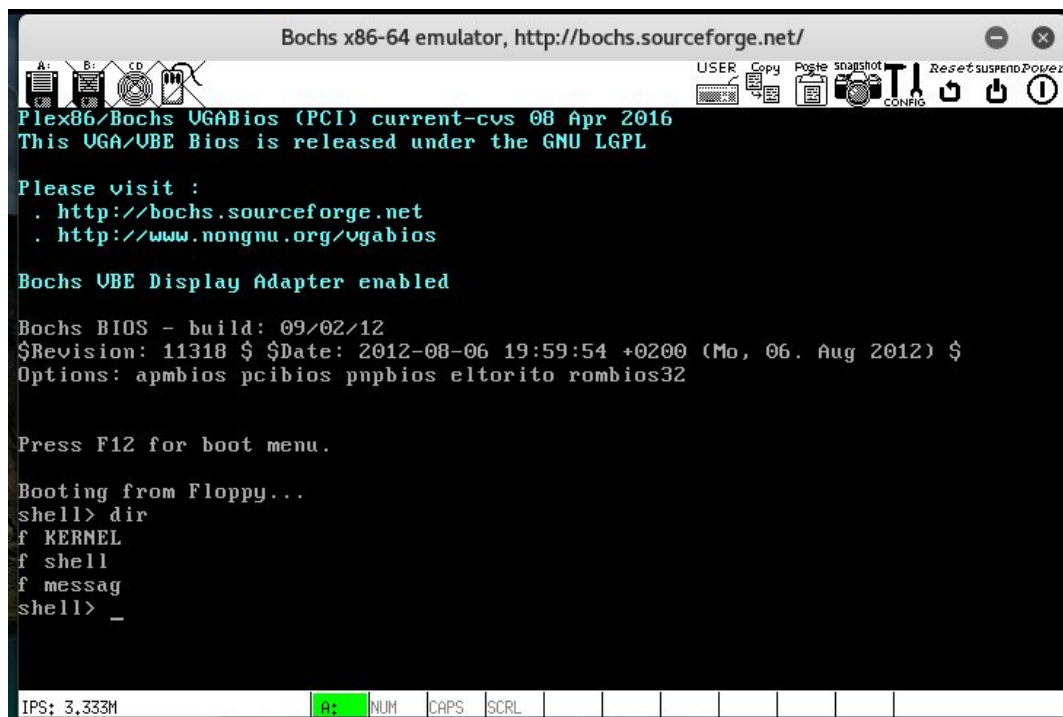
```
Other (3) - VMware Workstation 12 Player (Non-commercial use only)
File Virtual Machine Help

shell> dir
f KERNEL
f shell
f messag
d coba
shell> cd coba
shell> create Hello
Hello World!

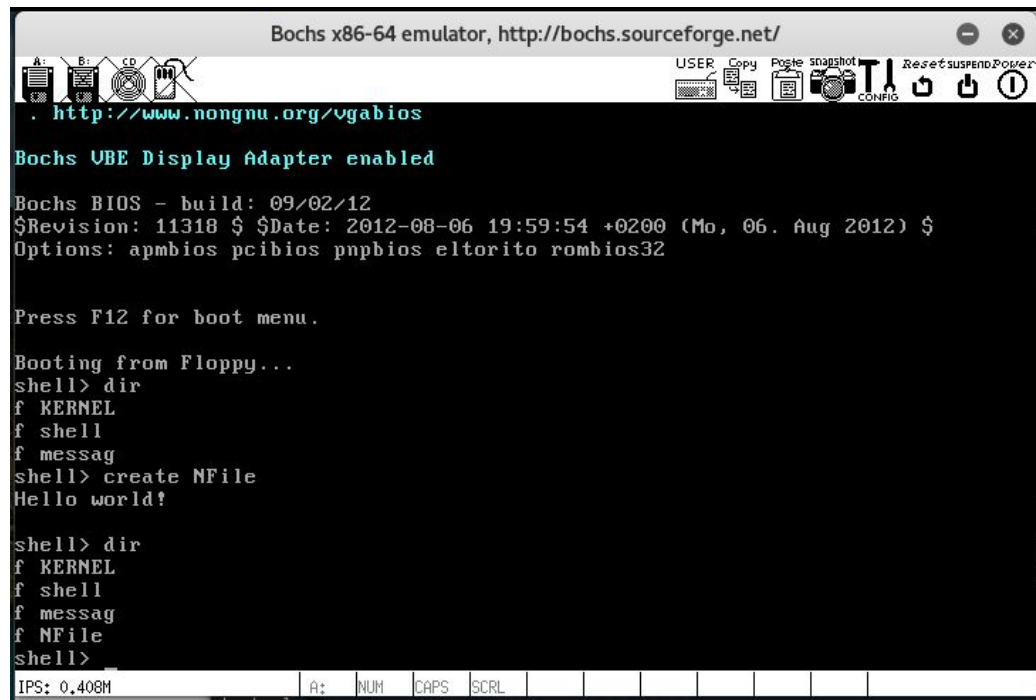
shell> dir
f Hello
shell> cd ..
shell> dir
f KERNEL
f shell
f messag
d coba
shell> delete coba
shell> dir
f KERNEL
f shell
f messag
shell> _

To grab input, press Ctrl+G
```

Gambar 6. P4\_extended\_kernel : membuat file dalam direktori, dan menghapus directory tersebut dengan command delete



Gambar 7. Screenshot saat OS dimulai di bosch dan menjalankan command dir untuk melihat konten dari OS tersebut, untuk P4\_extended\_kernel.



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
. http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

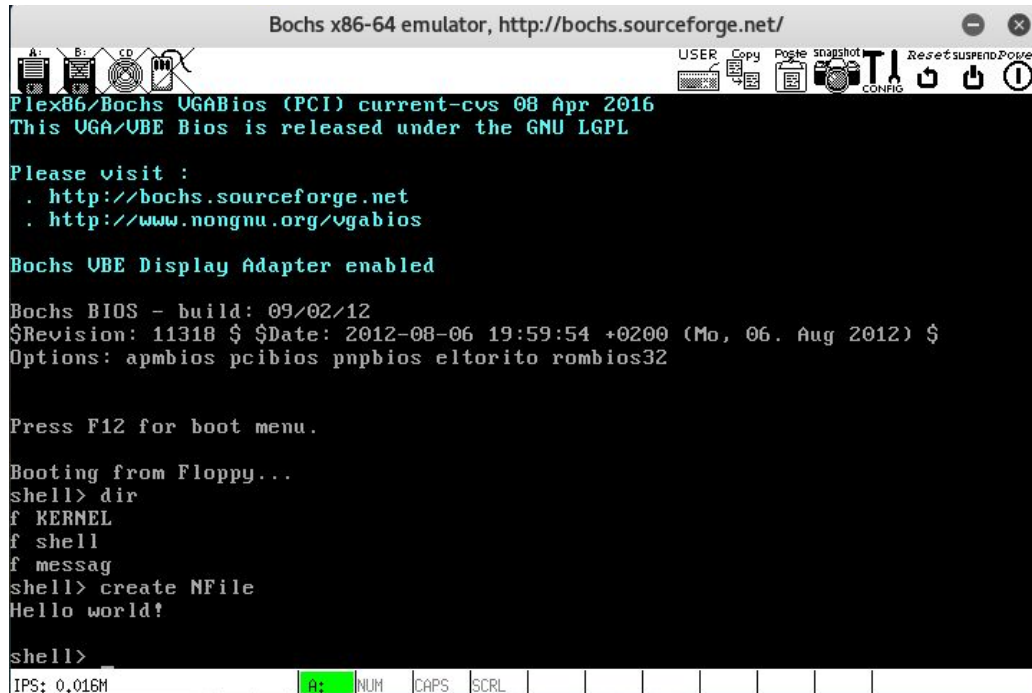
Bochs BIOS - build: 09/02/12
$Revision: 11318 $ $Date: 2012-08-06 19:59:54 +0200 (Mo, 06. Aug 2012) $
Options: apmbios pcibios pnpbios eltorito rombios32

Press F12 for boot menu.

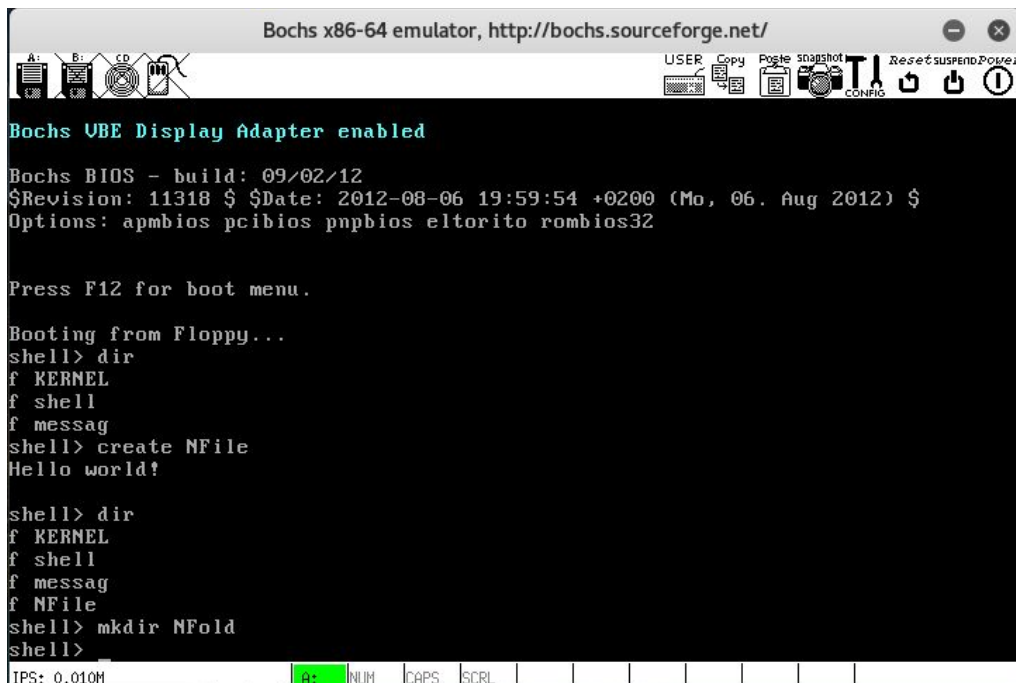
Booting from Floppy...
shell> dir
f KERNEL
f shell
f messag
shell> create NFile
Hello world!

shell> dir
f KERNEL
f shell
f messag
f NFile
shell>
```

Gambar 8. Screenshot saat OS menjalankan perintah create untuk membuat sebuah File baru dan berisi “Hello World!”, untuk P4\_extended\_kernel.



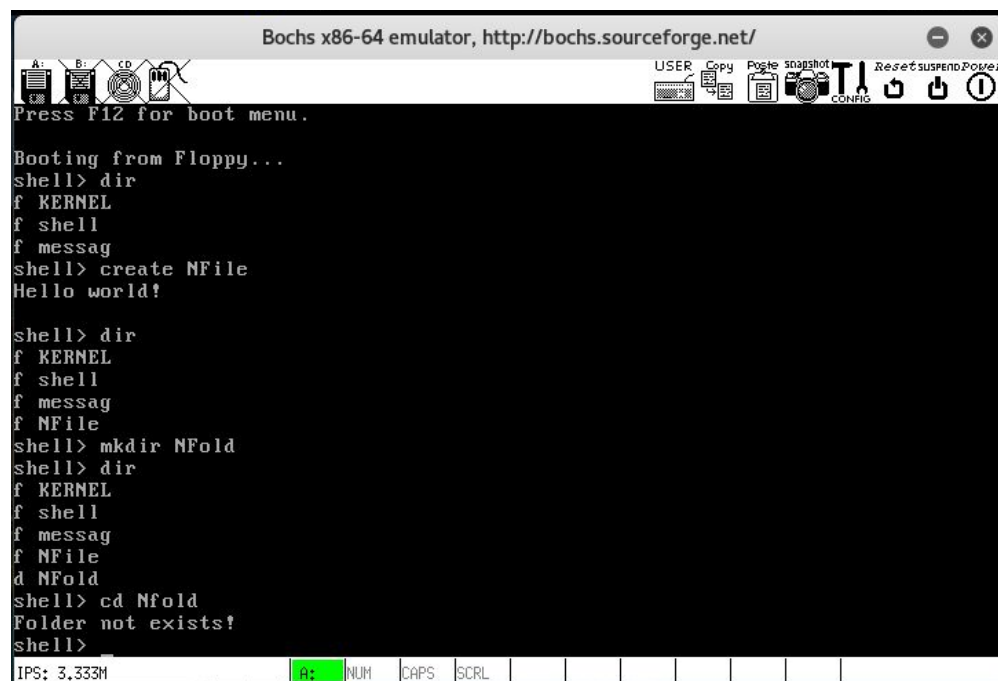
Gambar 9. Screenshot saat OS menjalankan perintah dir setelah menjalankan create NFile, untuk P4\_extended\_kernel.



Gambar 10. Screenshot saat OS menjalankan perintah mkdir untuk menciptakan sebuah folder baru, untuk P4\_extended\_kernel.



Gambar 11. Screenshot saat OS menjalankan command dir setelah menjalankan perintah mkdir, untuk P4\_extended\_kernel.



Gambar 12. Screenshot saat OS menjalankan perintah cd (*change directory*) namun untuk folder yang tidak ada, untuk P4\_extended\_kernel.

```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
A: B: CD
USER Copy Paste Snapshot CONFIG Reset Suspend Power
Booting from Floppy...
shell> dir
f KERNEL
f shell
f messag
shell> create NFile
Hello world!

shell> dir
f KERNEL
f shell
f messag
f NFile
shell> mkdir Nfold
shell> dir
f KERNEL
f shell
f messag
f NFile
d Nfold
shell> cd Nfold
Folder not exists!
shell> cd Nfold
shell> dir
shell>
```

IPS: 3,333M

A:	NUM	CAPS	SCRL																
----	-----	------	------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Gambar 13. Screenshot saat OS menjalankan perintah `cd` (*change directory*) untuk folder yang telah dibuat, untuk P4\_extended\_kernel.

#### D. Pembagian Tugas

NIM	Hal yang dikerjakan	Persentase
13515043	<ul style="list-style-type: none"><li>- Membuat laporan</li><li>- Menjawab soal pada laporan</li><li>- Mengerjakan sebagian p4_kernel</li></ul>	30%
13515097	<ul style="list-style-type: none"><li>- Mengerjakan p2_kernel, p3_kernel, p4_kernel, dan p4_extended_kernel</li><li>- Membuat laporan</li></ul>	40%
13515100	<ul style="list-style-type: none"><li>- Membuat laporan</li><li>- Mengerjakan bonus</li><li>- Mengerjakan sebagian p2_kernel dan p3_kernel</li></ul>	30%

#### E. Feedback

Materi tubes yang diberikan sangat membantu mahasiswa untuk dapat mengetahui sistem OS lebih dalam lagi, khususnya yang ingin menciptakan OS sendiri.