OPTIMASI KERNEL LINUX PADA ARSITEKTUR PROSESOR TI OMAP 3430

Aldyth M[1], A Subkhan KH, ST [2]
[1]Mahasiswa Politeknik Elektronika Negeri Surabaya, Jurusan Teknik Telekomunikasi
[2] Dosen Politeknik Elektronika Negeri Surabaya Institut Teknologi Sepuluh Nopember
Kampus ITS, Surabaya 60111
e-mail : bsdraisefromhell@gmail.com e-mail : subhankh@eepis-its.edu,

Abstrak

Pemakaian *mobile phones* dewasa ini semakin berkembang, tidak hanya untuk sekedar pemakaian untuk keperluan mengirim pesan atau telepon tetapi perkembangan pemakaian sudah mengarah kepada menjalankan fungsi *multimedia*. Tetapi pada kenyataannya masih banyak sekali sistem operasi yang tidak memiliki dukungan secara penuh terhadap fiturfitur baru yang telah dimiliki oleh *mobile phones*. Sehingga menyebabkan banyak fitur yang tidak dapat dioptimalkan oleh *user* dan menjadi suatu kerugian tersendiri. Pada proyek akhir ini akan dilakukan pengembangan sistem operasi berbasis linux dan *JVM* dengan harapan dapat menghasilkan sistem operasi yang handal dan memiliki dukungan penuh terhadap fitur-fitur terbaru khususnya fitur *multimedia*. Dengan melakukan perbandingan terhadap sistem operasi yang telah ada sebelumnya yaitu *Android,* akan terlihat bahwa sistem operasi yang dikembangkan telah layak atau tidak untuk menjalankan fiturfitur multimedia.

Kata kunci : sistem operasi, kernel linux, JVM

1. Pendahuluan

Semakin berkembangnya dunia teknologi khususnya pada bidang *ICT(Information and Communication Technology)* maka perkembangan disertai dengan semakin mutakhirnya *hardware* atau perangkat keras yang beredar di pasaran. Perkembangan dari perangkat keras ini menjadikan barang hasil teknologi tersebut menjadi memiliki berbagai fungsi yang serbaguna. Fitur-fitur yang menjadi andalan para *vendor* pada saat ini lebih mengarah kepada fitur pengembangan *multimedia*. Sebagai contoh seperti pemutar musik, pemutar film, akses *audio* dan *video streaming* serta berbagai macam dukungan *multimedia* lainnya. Dari hal tersebut maka diperlukan juga pengembangan pada sisi sistem operasi pada *mobile phones* tersebut, karena perkembangan *hardware* yang begitu pesat juga membuat para pengembang sistem operasi berlomba-lomba untuk membuat sistem operasi yang stabil dan mudah digunakan oleh pemakai. Karena itu akhirnya banyak variasi sistem operasi yang beredar di masyarakat baik dari yang berbayar(*proprietary*) atau bebas*(free)*. Pengembangan pada sistem operasi akan berdampak pada pengembangan aplikasi-aplikasi pada peralatan tersebut. Pengembangan sistem operasi yang akan dijalankan pada *mobile phones* lebih pada pengembangan berbasis *multimedia*, dimana sistem operasi tersebut memiliki fasilitas yang dapat mengoptimalkan fungsi dari *hardware* yang memiliki dukungan terhadap fitur *multimedia*.

Dalam proyek akhir ini akan dilakukan optimasi pada kernel linux dengan tujuan untuk mengoptimalkan sistem operasi yang memiliki dukungan tinggi pada fitur-fitur yang telah disebutkan sebelumnya khususnya pada bidang *multimedia* dari *OMAP 3430* dan mengoptimalkan fungsi dari *JRE(Java Runtime Environment)* sehingga akan mudah dikembangkan aplikasi-aplikasi yang berjalan di atasnya. Optimasi pada kernel linux meliputi *interprocess communication, scheduler,* dan m*emory management* dengan harapan *JRE* dan dukungan terhadap fitur *multimedia*.

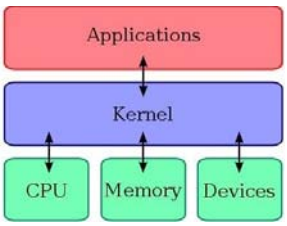
2. Teori Penunjang

Bootloader

Bootloader adalah bagian dari sistem operasi yang melakukan *booting*, serta pengambilalihan tugas dari *BIOS* ke sistem operasi. Selanjutnya *bootloader* akan menjalankan kernel sehingga tugas untuk menjalankan fungsi-fungsi berikutnya dilakukan oleh kernel.

Kernel

Kernel merupakan jantung sistem operasi, segala aktifitas yang dilakukan oleh sistem operasi akan dikontrol oleh kernel. Pada proyek akhir ini menitik beratkan pada optimasi sisi kernel dikarenakan kernel memiliki fungsi yang sangat vital dalam pengembangan suatu sistem operasi. Sehingga optimasi pada sisi kernel ini akan membawa pengaruh yang sangat besar pada pengembangan sistem operasi ke depannya dan memungkinkan akan semakin banyak aplikasi yang dikembangkan untuk memenuhi kebutuhan *user*.

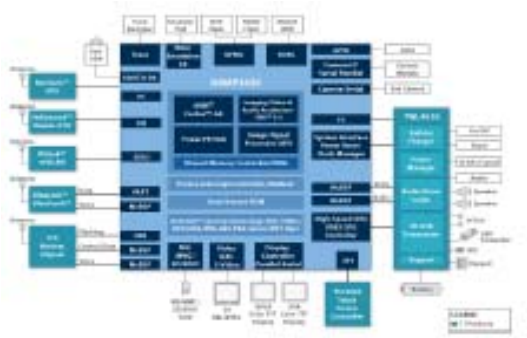


Gambar 1. Layout Kernel

Pada gambar diatas terlihat bahwa kernel adalah merupakan *layer* antara sisi *user* dan *hardware* dimana *user* berinteraksi dengan aplikasi. Kernel memiliki fungsi utama yaitu mengatur proses *input* dan *output* dimana proses itu memiliki bagian-bagian seperti berikut[8] :

1. Manajemen Proses
2. Manajemen Memori
3. Manajemen *Device*(perangkat)
4. *System Calls*
5. *File Subsystem*

OMAP 3430



Gambar 3. **Arsitektur OMAP 3430**

OMAP(Open Multimedia Application Platform) adalah termasuk dalam kategori *proprietary microprocessors*, yaitu mikroprosesor yang berbayar dan dilindungi oleh hak cipta. Mikroprosesor ini memiliki kemampuan *portable* dan pengembangan aplikasi *multimedia* pada perangkat mobile dan dikembangkan oleh *Texas Instrument*. Ada beberapa jenis yang termasuk dalam tipe prosesor keluarga *OMAP* ini yaitu *OMAP 34xx, 35xx, 36,* dan *44xx*. Beberapa tipe prosesor *OMAP* memuat arsitektur *dual core* dimana terdapat host prosesor *ARM* dan satu atau lebih prosesor *DSP*.

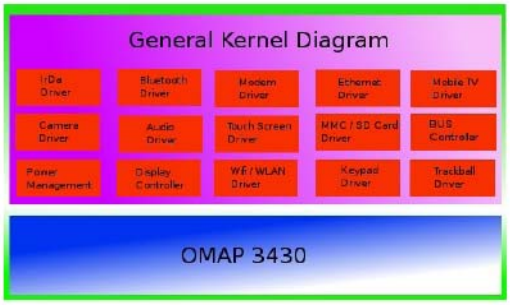
3. Perancangan

Rancangan Sistem

Sebagai langkah pertama, dilakukan pembuatan rancangan kernel terlebih dahulu supaya dapat mengerti modul apa saja yang akan dimasukkan pada kernel. Hal ini

memegang peranan sangat penting karena sesuai dengan teori penunjang yang telah disebutkan pada bab sebelumnya, kernel memiliki peran sebagai inti dari sistem operasi yang mengendalikan segala proses yang berjalan di sistem operasi tersebut. Rancangan kernel ini akan berpengaruh performa kernel linux tersebut karena sifat dari kernel linux yang termasuk pada *monolithic kernel* maka jika terdapat satu modul yang mengalami kerusakan akan berdampak kerusakan juga pada kernel linux secara keseluruhan. Proses ini akan dilakukan sebelum melakukan kompilasi.

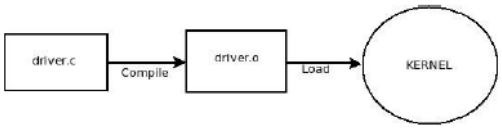
Pada gambar tersebut terlihat kernel linux yang akan di*download* pada *OMAP* memiliki modul-modul untuk mengenali *device* yang dimiliki oleh *OMAP*. Modul tersebut merupakan *driver* yang berfungsi sebagai abstraksi *device* untuk mengoptimalkan kemampuan *device* sehingga fitur *multimedia* dapat dikembangkan untuk menjadi lebih baik performanya.



Gambar 4. Rancangan Kernel

Karena kernel linux yang bersifat monolitik, maka setiap *driver yang* berfungsi untuk mengenali *hardware* akan dianggap sebagai suatu modul. Tetapi tidak semua modul adalah *driver*, modul juga dapat merupakan suatu *dependencies* / ketergantungan dari suatu paket / aplikasi yang berjalan pada sistem operasi. Pada proyek akhir ini, dilakukan pemilihan modul mana saja yang dibutuhkan oleh sistem operasi sehingga minimal dapat berjalan dengan baik dan fitur utama dari *omap 3430 zoom1* dapat digunakan dengan optimal.

Seperti yang terdapat pada gambar tersebut, maka beberapa *driver* utama akan dimasukkan pada kernel dan akan diuji kestabilannya dan keamanannya. Disini, *driver* dari linux dibangun menggunakan pemrograman bahasa C yang nantinya akan dilakukan kompilasi dan akan dilakukan *load* ke kernel sehingga *driver* nantinya dapat dijalankan sehingga *hardware* dapat dikenali.



Gambar 5. Rancangan Kernel

4. Pengujian dan Analisa

4.1 Pengujian Driver

1.Driver LCD

Pada pengujian *driver lcd* berhasil dijalankan oleh kernel linux. Kernel linux mengenali sebagai *omap display hardware.* Seperti yang ditunjukkan oleh proses *booting* sebagai berikut,

```
<6>OMAP Display hardware version 2.0
<6><6>TWL4030: Driver registration complete.
<7>omap2_disp_outLCD panel 480x640
<7>omap2_disp_outTV 640x480 interlaced
<6>omap24xxfb: Options "<NULL>"
<7>Frame buffer -- address = 0xfe200000
<7>omap24xxfb_get_flipbuffers::flipping buffers =3
Console: switching to colour frame buffer device 60x40
```

Pada *board Zoom OMAP 3430*, untuk *driver* yang mengenali *device* pada *board* adalah driver yang terletak pada struktur direktori */drivers/video/omap* yaitu *driver* omapfb. Pada bagian ini, *driver* yang terdapat pada direktori tersebut memiliki berbagai fungsi yang berbeda. Pada *source code omap_fb.c* berfungsi untuk mengatur *framebuffer* dari *board*, dimana *framebuffer* ini berfungsi sebagai output video yang mendorong tampilan video dari *buffer* suatu memori berisi *frame* data yang lengkap. Sedangkan pada *file omap_disp_out.c* berisi *display out* yang bertugas untuk memegang kendali terhadap *lcd* pada *board omap zoom*.

Gambar 6. LCD

2. *Driver Touchscreen*

Driver touchscreen ini terletak pada struktur direktori kernel */drivers/input/touchscreen/omap.* Pada *board Zoom OMAP 3430, touchscreen* digunakan sebagai salah satu *input* interaksi *user* dengan *board*.

```
<6>input: ADS784x Touchscreen as /class/input/input2
input: ADS784x Touchscreen as /class/input/input2
```

Ketika proses *booting, touchscreen input* didefinisikan sebagai */class/input/input2*. Karena pada linux, setiap bentuk *input device* akan dikenali sebagai suatu *device* dan didefinisikan pada direktori */dev*. Sebagai *file* fungsional akan didefinisikan dalam direktori */class/* yang merupakan sub direktori dari */sys*. Sehingga dalam proses *booting* akan dikenali sebagai suatu kernel *event* dan akan melakukan suatu inisialisasi dan dapat difungsikan.

3. *Driver twl4030*

Twl4030 ini memiliki peranan penting dalam berjalannya suatu sistem operasi untuk mengenali *hardware* yang ada. Pada *board Zoom OMAP 3430, driver* ini berfungsi untuk menangani fungsi dari *keypad, battery charger, audio / video codec, power manager. Twl4030* ini adalah *all-in-one audio* dan *power management* pada prosesor *omap*. Pada *driver* ini, masih banyak kendala yang penulis temui. Sebagai contoh pada *keypad driver*, sudah dapat dikenali tetapi masih banyak kesalahan sehingga tidak dapat berjalan dengan baik

```
<6>input: omap_twl4030keypad as /class/input/input0
input: omap_twl4030keypad as /class/input/input0
```

Ketika proses *booting* terjadi, *keypad* dapat dikenali oleh kernel, dan pada kernel *event* tercantum bahwa *keypad* diinisialisasikan pada */class/input/input0*.

Permasalahan terjadi karena *matrix* dari *keypad* yang tidak dapat terjadi dengan baik. Sehingga *keypad* tidak dapat dioptimalkan dengan baik. Hal tersebut menyebabkan *keypad* untuk saat ini belum dapat difungsikan, sehingga *input* hanya dari *touchscreen*.

Driver twl4030 dapat juga sebagai pengatur *battery charger*. Pada bagian tersebut, secara *default* sudah dapat dikenali oleh kernel linux sehingga tidak memerlukan kustomisasi ulang. Pada *driver audio/video codecs*, menggunakan *alsa(advanced linux sound architecture). Alsa* adalah *library* bagian dari komponen kernel linux, *alsa* berfungsi untuk mengenali *soundcard* pada suatu perangkat tertentu.

```
<6>TWL4030 Audio Codec init
TWL4030 Audio Codec init
<6>asoc: twl4030 <-> omap-mcbsp-dai-0 mapping ok
asoc: twl4030 <-> omap-mcbsp-dai-0 mapping ok
<6>ALSA device list:
ALSA device list:
<6>  #0: LDP (twl4030)
  #0: LDP (twl4030)
```

Permasalahan *driver alsa* adalah tidak dapat mengenali *sound card* yang terdapat pada *board Zoom OMAP 3430*. Sehingga sampai pada saat ini, untuk rilis kernel terbaru saja masih banyak *bug* pada *sound*.

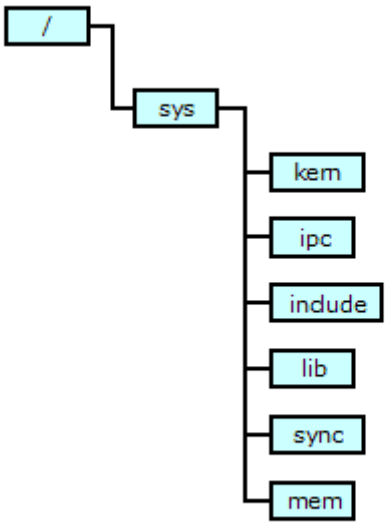
4.2 Pengujian Antarmuka

Supaya kernel dapat diketahui berjalan dengan baik, maka perlu dilakukan suatu pengujian dengan tampilan antarmuka. Seperti telah dijelaskan sebelumnya, *driver* dimuat pada kernel dan diuji dengan

melakukan *booting* pada *board Zoom OMAP 3430*. Ketika terjadi *booting*, maka kernel akan mengeluarkan suatu *event* dimana *event* tersebut memuat tentang aktifitas kernel sebelum masuk pada terminal / antarmuka. Telah dijelaskan proses kompilasi pada *android* dan setelah itu *filesystem android* disalin pada *microsd card* sebagai *storage media*. Disini, kernel harus mengetahui posisi direktori *root* supaya dapat masuk terminal / antarmuka.

1. Kernel

Sebelum kernel diuji pada *board Zoom OMAP 3430*, langkah awal adalah melakukan konfigurasi terlebih dahulu pada kernel. Konfigurasi disini dilakukan sebelum proses kompilasi, hal tersebut harus dilakukan mengingat kernel dapat digunakan untuk berbagai macam kebutuhan sehingga pada nantinya kernel dapat disesuaikan dengan kebutuhan.



Gambar 4.1 Struktur direktori kernel

Sebelumnya telah dijelaskan langkah membangun kernel dengan menggunakan konfigurasi *omap3430labrador_config*. File ini berisi konfigurasi kernel supaya ketika selesai dilakukan proses *cross-compiling* dapat dijalankan pada *board Zoom OMAP 3430*.

Jika ingin hemat dalam melakukan kompilasi, maka pilih *driver* yang sekiranya cocok dengan *hardware* yang ada. Karena dalam *board Zoom OMAP 3430*, sangat kecil kemungkinan untuk menambah *add-on hardware*. Disini sebagai contoh penulis untuk *driver Ethernet* menggunakan *SMC* dan untuk prosesor menggunakan *arm cortex a-8*. Sehingga hasil *image* yang dihasilkan cukup minim yaitu berukuran 2.0MB. Supaya *android* dapat berjalan maka perlu ditambahkan *driver binder* untuk menjalankan optimasi fungsi *IPC(Inter Process Communications).*

2.Android

Setelah kernel berhasil dikonfigurasi maka langkah selanjutnya adalah mencoba untuk melakukan sinkronisasi dengan tampilan antarmuka *android*. Disini digunakan *android* rilis RLS25.7 dengan kernel 2.6.27-24. Tetapi selama ini, penulis masih banyak menemukan kendala dengan *android*. Kendala utama pada tidak dapat memasuki *desktop android*, ketika akan memasuki *desktop* selalu secara tiba-tiba meminta untuk *shutdown.* Ketika proses perpindahan dari kernel menuju *android* maka *file* pertama yang akan dieksekusi adalah *init.rc.*

Daftar Pustaka :

[1]https://omapzoom.org/gf/project/omapzoom/wiki/pagename=BootingAndFlashing
[2]http://en.wikipedia.org/wiki/Kernel_(computing)
[3]http://en.wikipedia.org/wiki/Unix_shell
[4]http://en.wikipedia.org/wiki/Java_Virtual_Machine
[5] Bill Venners, *Inside The Java Virtual Machine*, Edisi Pertama, McGrawHill, USA, 1997
[6] Masyarakat Digital Gotong Royong, *Pengantar Sistem Operasi*, Edisi Kedua,

Masyarakat Digital Gotong Royong, Jakarta, 2008
[7] Budi Halus Santosa, *Perancangan Sistem Operasi*, Edisi Pertama, Andi Publisher, 2006
[8] Abraham Silberschatz and Peter Baer Galvin, *Operating System Concepts*, AddisonWesley, USA, 1998
[9] Wolfgang Mauerer, *Professional Linux Kernel Architecture*, Edisi Pertama, John Wiley Publishing, USA, 2008
[10] Dominic Giampaolo, *Practical File System Design with the Be File System,*