MAKALAH TUGAS
SISTEM OPERASI
KONSEP PROSES, *THREADS* DAN KERNEL
SERTA APLIKASINYA PADA WINDOWS 10 DAN ANDROID OS

Nama : Wildan Fajar Purnomo
NIM : 17/415241/TK/46530

DEPARTEMEN TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2018

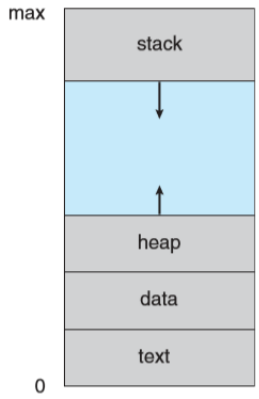
Daftar Isi

BAB 1

Konsep Proses, *Thread* dan Kernel

A. Konsep Proses

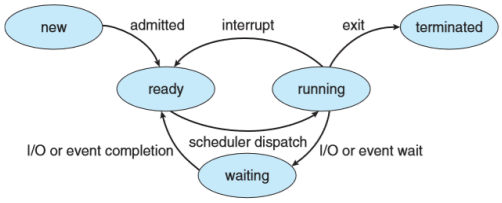
Proses adalah pengerjaan program pada OS. Umumnya setiap *device* memiliki berbagai aplikasi atau program. Ketika *user* menjalankan salah satu atau beberapa program pada saat itulah proses akan tercipta dan mengerjakan komponen program sehingga *user* dapat menggunakan program tersebut. Dalam menjalankan program, proses tidak hanya berupa kode tetapi juga alamat program dalam ROM (ditunjukkan oleh *program counter*). Secara umum proses juga meliputi *stack*, *data section* dan *heap*. *Stack* merupakan tempat untuk menyimpan data sementara. *Data section* merupakan wadah variabel global. *Heap* merupakan struktur memori dinamis dalam jalannya proses.



Gambar 1. Struktur proses dalam Memori

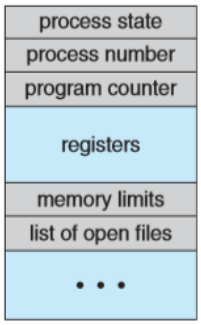
Sebuah program dapat diasosiasi oleh satu atau lebih proses. Contoh asosiasi oleh lebih dari satu proses adalah misalnya beberapa *users* menjalankan beberapa salinan dari *mail program*. Dalam hal ini satu *mail program* terasosiasi oleh beberapa proses dari pengguna-pengguna aktif. Jika demikian maka proses-proses yang berjalan dianggap sebagai proses-proses yang terpisah. Meskipun mereka berada pada program yang serupa bahkan sama akan tetapi penggunaan *data section*, *stack* dan *heap* masing-masing proses akan berbeda, karenanya dianggap sebagai proses terpisah.

Proses memiliki lima kondisi ketika berjalan. Kelima kondisi tersebut adalah *new*, *running*, *waiting*, *ready* dan *terminated*. *New* adalah kondisi ketika proses baru saja diciptakan, umumnya terjadi ketika suatu aplikasi pertama kali dijalankan. *Running* adalah ketika proses telah mengeksekusi instruksi atau komponen aplikasi. *Waiting* adalah ketika proses menunggu *event* tertentu seperti contohnya menunggu finalisasi I/O. *Ready* adalah ketika proses menunggu tugas dari prosesor. Akhirnya, *terminated* adalah ketika proses menyelesaikan eksekusi dan dihentikan.



Gambar 2. Diagram kondisi proses

Dalam OS, setiap proses direpresentasikan dalam *process control block* atau lebih dikenal sebagai PCB. PCB meliputi lima proses spesifik diantaranya *process state*, *program counter* , *CPU registers*, *CPU – scheduling information*, *memory management information*, *accounting information* dan *I/O status information*.

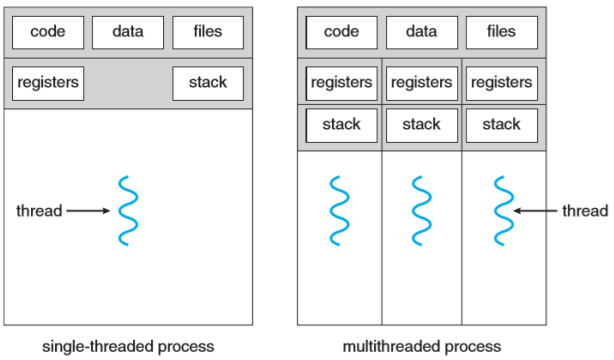


Gambar 3. *Process control block*

B. Konsep *Threads*

Thread merupakan entitas dalam proses yang dapat dijadwalkan untuk melakukan eksekusi. Dalam satu proses, beberapa *thread* membagi *virtual address space* dan *resource* dari sistem. Setiap *thread* memiliki *exception handler* (penanganan atau alternatif jika terjadi sesuatu yang tidak diharapkan selama proses), prioritas penjadwalan, *local storage*, ID.

Membicarakan *thread* tidak akan lepas dari konsep *single-threaded* dan *multithreaded*. *Single-threaded* adalah konsep dimana satu proses dipegang oleh satu *thread*. *Multithreaded* adalah konsep dimana satu proses dapat dipegang lebih oleh satu *thread*. Saat ini konsep *multithreaded* semakin banyak digunakan.



Gambar 4. Ilustrasi *single-threaded* dan *multithreaded*

C. Konsep Kernel

Kernel merupakan komponen utama dalam OS. Singkatnya, kernel menjembatani kerja komponen utama sistem computer yaitu prosesor (CPU), memori dan *input/output devices*. Dalam perannya, kernel menggunakan *interprocess communication* dan *system calls*. Secara umum kernel bertanggung jawab atas empat hal : manajemen proses untuk eksekusi aplikasi, manajemen memori dan pengaturan *resources*, management *devices* dan kontrol terhadap *system call*.

Beberapa contoh pekerjaan dari kernel adalah menyediakan *memory address space* (alamat memori) untuk aplikasi, mengisi memori dengan kode aplikasi dan menyediakan memori *stack* untuk menunjang jalannya program dalam OS. Kernel sendiri memiliki lima tipe yaitu monolitik, *microkernel* , *exokernel, hybrid* kernel, *nanokernel*.

BAB 2

Proses, Threads dan Kernel Pada Windows 10 OS

A. Konsep Proses dan *Threading* Pada Windows 10

Pada Windows, proses dan *thread* masing-masing diberikan ID (*identifier*) sehingga pasangan proses dan *thread* dapat terlihat jelas. Dalam *threading*, Windows 10 menggunakan konsep *multithreading* yaitu dimana satu aplikasi atau satu proses dapat memiliki lebih dari satu *thread*. Meskipun demikian, *performance*-nya tergantung dimana OS Windows berjalan. Pada *device* dengan satu prosesor, meskipun setiap aplikasi dapat memiliki banyak *thread* namun hanya satu yang dapat dijalankan pada satu waktu. Konsep *multithreading* pun tidak dapat sepenuhnya teraplikaskan. Pada *device* dengan lebih dari satu prosesor, konsep *multithreading* dapat dijalankan secara sepenuhnya karena *thread-thread* dapat berjalan bersamaan pada prosesor yang berbeda-beda.

Salah satu masalah yang dapat muncul akibat *multi threading* tersebut adalah jika sebuah atau beberapa *thread* mencoba menggunakan *resource* yang sama dalam satu waktu. Untuk mencegah hal ini terjadi, Windows memiliki teknik yang disebut sinkronisasi.

B. Konsep Kernel Pada Windows 10

Secara umum kernel pada Windows 10 merupakan *hybrid kernel*. Artinya kernel Windows memiliki keunggulan dari arsitektur *microkernel* dan monolitik. Sebenarnya kernel pada Windows 10 merupakan turunan dari kernel yang digunakan pada Windows NT, yaitu NT kernel. Jadi keseluruhan kernel Windows 10 mirip dengan NT kernel.

Windows memiliki mode yang disebut sebagai kernel mode. Dengan kernel mode *user* memiliki akses yang lebih luas terhadap *hardware* PC. Kernel mode merupakan *manager* untuk banyak hal diantaranya *object manager*, *memory manager*, *process and thread manager*, *I/O manager*, *power manager*, *configuration manager*, dan masih banyak lagi.

BAB 3

Proses, Threads dan Kernel Pada Android OS

A. Overview

Pada OS Android, proses yang digunakan untuk eksekusi komponen aplikasi adalah proses Linux. Ketika suatu aplikasi dijalankan, normalnya sebuah proses Linux akan dipanggil dan semua komponen aplikasi akan dijalankan dalam satu *main thread*. Jika ternyata sebuah proses sudah ada tanpa harus dipanggil, maka tentu saja komponen program akan dijalankan pada proses dan *thread* yang telah tersedia tersebut.

B. Konsep Siklus Hidup Proses dan Aplikasi Pada OS Android

Seperti yang telah dipaparkan sebelumnya bahwa kebanyakan aplikasi Android berjalan pada proses Linux. Tentu saja proses ini akan terus berjalan selama masih terdapat kode yang dapat dijalankan. Proses berhenti ketika penggunannya sudah tidak digunakan (misalnya aplikasi tidak lagi dijalankan pada proses tersebut) dan ketika system membutuhkan memori untuk menjalankan aplikasi lainnya.

Bagaimana siklus hidup proses pada Android? Rupanya siklus hidup proses tidak bergantung semata-mata pada aplikasi itu sendiri. Siklus hidup pada Android ditentukan oleh system dengan memperhatikan tiga hal yaitu bagaimana komponen-komponen aplikasi saling bekerja, seberapa penting aplikasi yang bekerja dan kesediaan memori pada system secara keseluruhan.

Memperhatikan pertimbangan ketiga yaitu kesediaan memori pada system, ketika system mengalami *low memory* maka harus ada proses yang dihentikan (*killed*). Jika pada saat *low memory* terdapat beberapa proses yang sedang berjalan, Android akan mengelompokkan proses-proses menjadi sebuah hierarki prioritas untuk menentukan proses mana yang sebaiknya dihentikan. Berikut kelompok proses dimulai dari proses yang paling penting.

1. *Foreground Process*

Foreground process merupakan proses dengan prioritas tertinggi. Proses ini mengerjakan program yang sedang berinteraksi langsung dengan *user* pada saat tersebut. Tidak banyak proses yang dapat dikelompokkan sebagai *foreground*. Proses ini dianggap amat penting dan dihentikan hanya apabila ketersediaan memori sangat rendah, bahkan terlalu rendah hingga *foreground* sendiri tidak dapat berjalan.

2. *Visible Process*

Visible process memiliki prioritas lebih rendah dibandingkan *foreground*. Jujur saya sendiri kesulitan untuk memahami proses ini. Sejauh yang saya tangkap, proses ini mengerjakan program yang terlihat oleh *user*, akan tetapi *user* tidak sedang berinteraksi dengan program tersebut. Lebih sederhana lagi program tersebut tidak sampai berada di level *foreground*, hanya eksistensinya disadari oleh *user*. Proses ini terpaksa dihentikan jika memori sudah sangat terbatas hingga hanya dapat digunakan sepenuhnya untuk menunjang *foreground process*.

3. *Service Process*

Sesuai dengan namanya, proses ini menjalankan program *Service*. Secara teknis, *service* dijalankan dengan *method* **startService()**. Salah satu contoh dari *service process* adalah

background processing. Proses ini akan terus berjalan dan akan dihentikan paksa ketika memori hanya mampu menunjang *foreground* dan *visible service*.

4. *Cached Process*

Cached process merupakan proses yang kebetulan sedang tidak digunakan sehingga system pasti akan memutuskan untuk menghentikan proses ini terlebih dahulu dalam keadaan darurat memori. Jika keadaan memori system sedang normal, *cached process* berperan dalam manajemen memori dari Android. Sebenarnya, penghentian *cached process* tidak terbatas karena keadaan genting. System yang baik hendaknya rutin melakukan penghapusan *cached process* karena memang jika tidak dihentikan akan memakan memori yang sebenarnya bisa digunakan untuk proses yang lebih penting.

C. Konsep Thread Pada OS Android

Ketika sebuah aplikasi dijalankan, sebuah pasangan *process* dan *thread* dibuat. *Thread* yang dibuat ini dikenal sebagai *main thread*. Sesuai namanya, *thread* ini merupakan yang utama dan sangat diandalkan untuk menjalankan aplikasi. *Main thread* memegang peranan penting dalam *user interface* aplikasi. Contoh penggunaan *main thread* adalah sebagai tempat berjalannya *method* **onKeyDown()**. *Method* tersebut merupakan instruksi tentang bagaimana aplikasi berjalan jika sebuah tombol ditekan (di-*tap* oleh Android *user*). Karena perananannya dalam *user interface*, *main thread* terkadang juga disebut sebagai UI *thread*.

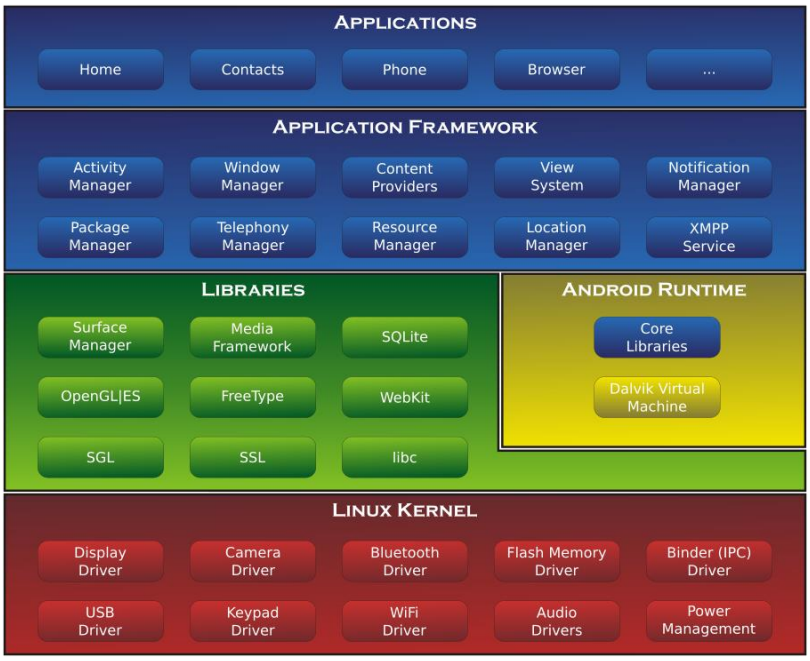
Android menggunakan model *single thread*. Android tidak otomatis menciptakan *thread* lagi seelah *main thread* dibuat. Hal ini menyebabkan *main thread* memiliki beban berat dalam menjalankan seluruh komponen aplikasi. Masalah dapat timbul jika aplikasi menuntut proses yang intensif. Jika semua proses berjalan pada *main thread* maka proses-proses yang rumit justru dapat mematikan *user interface*. Contoh proses rumit yang dimaksud adalah *network access* dan *database queries*. Ketika *user interface* mengalami galat, kondisi itulah yang dikenal sebagai ANR (*application not responding*) atau secara kasar "*hang*".

Terdapat salah satu cara untuk menghindari masalah diatas. Kita dapat menggunakan *worker threads*. Intinya adalah bagaimana kita mengalokasikan beberapa proses agar dijalankan di *worker threads* supaya beban pada *main thread* tidak terlalu berat.

D. Konsep Kernel Pada Android

Kernel pada Android merupakan Linux kernel, atau lebih tepatnya variasi dari Linux kernel. Linux kernel merupakan kernel *open source* yang bertipe monolitik. Saat ini Android menargetkan kernel versi 4.3, 4.9 atau 4.14 dari Linux Kernel. Karena merupakan variasi maka ada beberapa perbedaan dengan kernel Linux. Salah satu perbedaannya adalah Android secara *default* tidak memberikan akses kernel mode (*root*) seluas yang diberikan Linux kernel.

Pada keseluruhan arsitektur Android, Linux kernel memiliki peran dalam *display driver*, *camera driver*, *Bluetooth driver*, *flash memory driver*, *binder driver*, *USB driver*, *keypad driver*, *WiFi driver*, *audio drive* dan *power management*.



Gambar 5. Diagram arsitektur Android yang menunjukkan cakupan Linux Kernel

BAB 4

Referensi dan Daftar Pustaka

A. Referensi Buku

Silberschatz, A., Galvin, P.B. & Gagne, G., 2012. *Operating System Concepts* Ninth., Hoboken, NJ, United States of America: John Wiley & Sons.

B. Referensi Web

Lake, I., 2016. Who lives and who dies? Process priorities on Android. *Medium*. Tersedia pada laman: https://medium.com/androiddevelopers/who-lives-and-who-dies-process-priorities-on-android-cb151f39044f [Diakses September 13, 2018].

Anon, Understanding Android Application and Activity Lifecycles. *Drawing Graphics in C Sharp - Techotopia*. Tersedia pada laman: https://www.techotopia.com/index.php/Understanding_Android_Application_and_Activity_Lifecycles#Visible_Process [Diakses September 13, 2018].

Anon, 2018. Android (operating system). *Wikipedia*. Tersedia pada laman: https://en.wikipedia.org/wiki/Android_(operating_system) [Diakses September 13, 2018].

Anon, Processes and threads overview | Android Developers. *Android Developers*. Tersedia pada laman: https://developer.android.com/guide/components/processes-and-threads [Diakses September 13, 2018].

Anon, Processes and Application Lifecycle | Android Developers. *Android Developers*. Tersedia pada laman: https://developer.android.com/guide/components/activities/process-lifecycle [Diakses September 13, 2018].

windows-driver-content, Windows Kernel-Mode Process and Thread Manager. *Microsoft Docs*. Tersedia pada laman: https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/windows-kernel-mode-process-and-thread-manager [Diakses September 13, 2018].

Satran, M., About Processes and Threads. *Microsoft Docs*. Tersedia pada laman: https://docs.microsoft.com/en-gb/windows/desktop/ProcThread/about-processes-and-threads [Diakses September 13, 2018].

Anon, What is Kernel? - Definition from Techopedia. *Techopedia.com*. Tersedia pada laman: https://www.techopedia.com/definition/3277/kernel [Diakses September 13, 2018].

C. Sumber Gambar

Silberschatz, A., Galvin, P.B. & Gagne, G., 2012. *Operating System Concepts* Ninth., Hoboken, NJ, United States of America: John Wiley & Sons. (halaman 106, 108, 164)

Anon, 2018. Android (operating system). *Wikipedia*. Tersedia pada laman: https://en.wikipedia.org/wiki/Android_(operating_system)#/media/File:Android-System-Architecture.svg [Diakses September 13, 2018].