Penggunaan Algoritma *Depth-First-Search* dalam mendeteksi kemungkinan *Deadlock*

IF2211 Strategi Algoritma

Yoga Adrian Saputra - 13513030
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
Yoga.adrian@students.itb.ac.id

Kemajuan teknologi kita sering dianalogikan dengan banyaknya aplikasi yang bervariasi dengan berbagai macam fungsi yang sering membantu kita dalam kehidupan. Dalam menjalankan sebuah aplikasi, perangkat keras kita pasti memiliki suatu hal untuk mengatur keberjalanan sebuah aplikasi melalui sebuah proses. Pada Personal Computer atau laptop, sering kita dengar tentang Operating system.

Banyaknya aplikasi yang berjalan pada sebuah perangkat keras, terkadang bisa menyebabkan deadlock, sehingga terjadi "crash" pada Aplikasi kita. Disitulah peran Operating System.

Algoritma Depth-First-Search menjadi salah satu cara untuk mendeteksi adanya kesalahan seperti diatas.

Kata kunci : Aplikasi, Deadlock, Depth-First-Search, Deteksi.

I. PENDAHULUAN

Teknologi pada masa kini berkembang dengan sangat cepat. Hal tersebut sudah dibuktikan dengan spesifikasi-spesifikasi atau kemampuan yang dimiliki perangkat keras disekitar kita sangat luar biasa. Perkembangan teknologi pun tidak hanya dilihat dari perkembangan perangkat lunaknya saja. Namun juga perkembangan perangkat lunak yang luar biasa. Perkembangan perangkat lunak dan perkembangan perangkat keras ini harus seimbang.

Majunya perangkat lunak ini menyebabkan banyaknya aplikasi yang memiliki fitur yang sangat berguna dan bervariasi. Dengan kemampuan yang luar biasa dan bervariasi, pastinya implementasi kode dari aplikasi tersebut juga sangat kompleks. Implementasi yang kompleks terkadang membutuhan resource data yang juga banyak.

Jika hanya terdapat 1 aplikasi yang berjalan, maka kita tidak perlu takut adanya bug-bug yang aneh dalam implementasi aplikasi tersebut. Dalam arti lain, sebuah resource hanya dipakai oleh 1 proses tersebut.

Deadlock adalah kondisi dimana Aplikasi-aplikasi dalam perangkat keras tidak berjalan atau tidak menjankan aksinya. Hal tersebut terjadi karena adanya "saling tunggu" antara 2 atau lebih proses.

Seperti yang sudah dikatakan di atas, jika hanya terdapat satu aplikasi yang berjalan, maka tidak akan pernah terjadi Deadlock. Deadlock terjadi jika hanya terdapat 1 proses yang sedang bekerja dalam operating system. Nyatanya, Perangkat Keras zaman pasti bisa melakukan multitask

Disinilah peran Operating system ini, ia mengatur proses-proses yang berjalan dalam computer. Yaitu dari urutan proses-proses tersebut dikerjakan. Selain itu data atau resource resource yang diberikan dan di alokasi sebagaimana rupa hingga tidak terjadi deadlock pada sistem

Salah satu cara agar deadlock bisa terdeteksi adalah dengan mengimplementasi algoritma Depth-First-Search. Data penggunaan proses dan resource dalam sistem bisa digambarkan menjadi sebuah graf yang dinamakan "*Resource Allocation Graph*". Dari graf tersebut yang selalu berubah, maka akan dilakukan pengecekan secara berkala. Sehingga bila terjadi deadlock dalam sistem karena penjalanan sebuah proses baru, maka akan langsung terdeteksi apabila deadlock terjadi

II. DASAR TEORI

1. Proses
   a. Definisi

      Proses adalah bentuk fisik dari sebuah program komputer yang sedang dijalankan. Saat komputer berjalan, terdapat banyak proses yang berjalan secara bersamaan. Sebuah proses dibuat melalui system call create-process yang membentuk proses turunan (child process) yang dilakukan oleh proses induk (parent process). Proses turunan tersebut juga mampu membuat proses baru sehingga semua proses ini pada akhirnya membentuk pohon proses.

      Ketika sebuah proses dibuat, ada beberapa hal yang diberikan kepada proses. Yaitu sumber-daya seperti waktu CPU, memori, berkas, atau perangkat

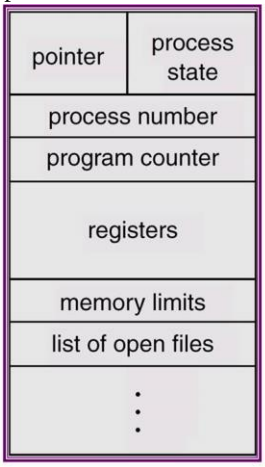
I/O. Sumber daya ini didapat diberikan langsung dari sistem operasi. Sumber daya tersebut juga bisa didapat dari proses induk yang membagi-bagikan sumber daya kepada setiap proses turunannnya atau proses turunan dan proses induk berbagi sumber-daya yang diberikan sistem operasi.

b. Informasi atau atribut sebuah proses

Secara keseluruhan, isi dari proses ialah:

- *Executable Machine Code* yang berasosiasi dengan pembuatan proses tersebut
- Memori, yang berisi kode eksekusi program, input dan output data, stack , dan sebuah heap untuk menyimpan kode program yang akan dijalankan saat eksekusi,
- File deskriptor dari resource yang diberikan kepada proses. Hal ini dibuat untuk keamanan atau agar proses lain tidak bisa membaca resource yang berada pada memori tanpa menggunakan file deskriptor tersebut. Pada sistem operasi Windows, file deskriptor ini disebut *handle*
- Atribut sekuriti. Yaitu atribut yang memberikan informasi tentang siapa pemilik proses ini dan hal akses-hak ases yang diberikan pada proses ini
- Status prosesor yang bertanggung jawab pada proses ini. Seperti alamat memori fisiknya atau isi register.

Hal-hal diatas ini disimpan pada *proses control block*. Setiap proses memiliki sebuah proses control block untuk menyimpan informasi tersebut.



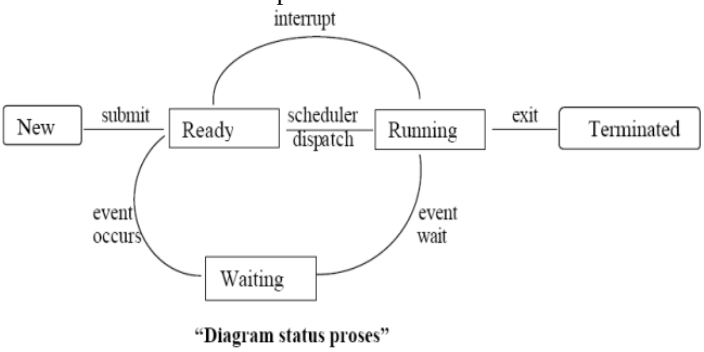
Gambar 1. Process Control Block sebuah Proses

Gambar diambil dari http://www.ustudy.in/node/844, diakses pada tanggal 5 Mei 2015 jam 05.09

c. State dalam Process

Selama masa hidupnya, sebuah proses memiliki banyak status. Yaitu:

- New : pembentukan suatu proses
- Running : instruksi-instruksi yang sedang dieksekusi
- Waiting : proses menunggu untuk beberapa event yang terjadi
- Ready : menunggu untuk dialirkan ke pemroses (processor)
- Terminated : proses telah selesai dieksekusi



Gambar 2. Diagram Status proses

Gambar diambil dari http://aliansyah.blog.upi.edu/2013/09/29/model-status-proses-pada-sistem-operasi/ diakses pada tanggal 5 Mei 2015 jam 05.15

d. Asdas

2. Resources

a. Definisi

Resource adalah komponen fisik atau virtual didalam sebuah sistem komputer. Daru bentuk fisik, resource biasanya berupa perangkat keras perangkat keras yang terhubung dalam komputer. Dalam bentuk Virtual bisa berupa file ,koneksi jaringan, atau area alokasi pada memori.

Pengaturan resource-resource ini dilakukan oleh resource management. Resource management ini mengatur hal-hal seperti:

- Resource Contention
  Yaitu pemberian resource pada salah satu proses ketika ada 2 atau lebih proses yang meminta resource saat bersamaan
- Resource Leaks
  Yaitu pengamanan resource yang telah dialokasikan pada suatu proses ketika proses itu telah selesai mengeksekusi kodenya.

b. Macam

- *File Handle*
- Koneksi Jaringan
- *Lock*
- *Perangkat Keras dari luar*
- Alokasi CPU
- Random Access Memory (RAM) dan Virtual Memory
- Tempat yang tersisa dalam hard disk
- Operasi masukan atau keluaran

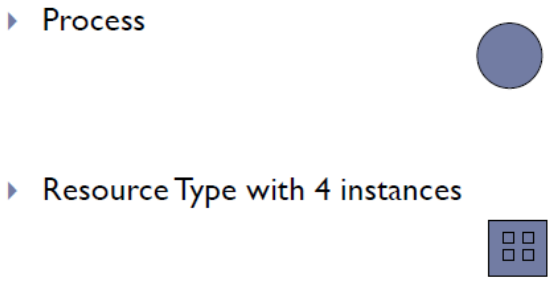
3. Penggambaran hubungan Proses dan Resources

(Resource-Allocation Graph)

a. Definisi
Resource-Allocation Graph adalah graf yang memberikan informasi tentang beberapa buah resource yang dialokasikan atau akan dialokasikan kepada proses tertentu

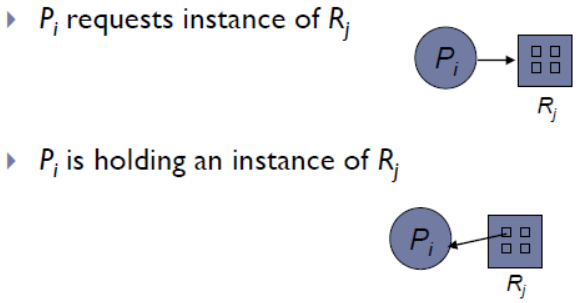
b. Informasi Komponen
Resource-Allocation Graph terdiri dari Komponen. Komponen-Komponen tersebut adalah:

- Node dibagi lagi menjadi 2, yaitu node untuk penggambaran proses(digambarkan dalam bentuk lingkaran dan/atau Karakter P) dan node untuk penggambaran resource(digambarkan dalam bentuk Persegi dan/atau Karakter R).



Gambar 3. Penggambaran Node pada Resource-Allocation Graph
Gambar diambil dari http://kuliah.informatika.org/ diakses pada tanggal 5 Mei 2015 jam 05.56

- Sisi juga dibagi menjadi 2. Yaitu permintaan (digambarkan dengan adanya panah dari proses ke resource) dan alokasi (digambarkan dengan adanya panah dari resource ke proses)

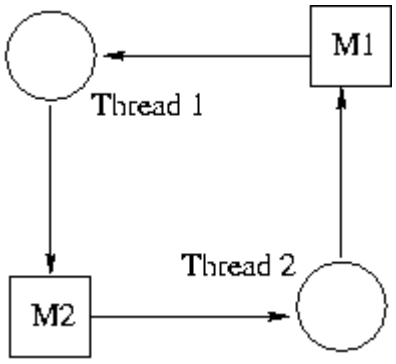


Gambar 4. Penggambaran Node pada Resource-Allocation Graph
Gambar diambil dari http://kuliah.informatika.org/ diakses pada tanggal 5 Mei 2015 jam 05.57

c. asa

4. Deadlock

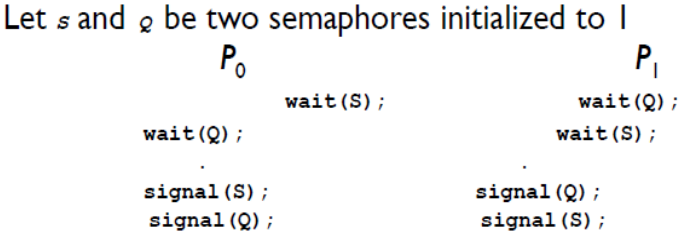
a. Definisi
Deadlock adalah kondisi dimana 2 atau lebih proses yang saling menghalangi satu proses sama lain untuk melanjutkan kode programnya. Biasanya hal ini terjadi karena adanya tabrakan dalam permintaan sebuah resource.



Gambar 5. Deadlock yang sederhana
Gambar diambil dari http://www.cs.rpi.edu/academics/courses/fall04/os/c10/ diakses pada tanggal 5 Mei 2015 jam 06.23

b. Penyebab
Deadlock bisa terjadi karena:

- Locks
Deadlock yang terjadi karena kesalahan pada implementasi Locks



Gambar 6. Contoh Implementasi Lock yang menyebabkan Deadlock
Gambar diambil dari http://kuliah.informatika.org/ diakses pada tanggal 5 Mei 2015 jam 06.32

Dari contoh Gambar 6, wait dan signal adalah bentuk implemetasi key yang berbentuk *semaphore*. Karena kedua prosess saling menunggu *signal* dari kunci Q dan S, sedangkan dari mereka tidak ada yang memberikan signal untuk kunci dari Q dan S, maka proses P0 dan P1 dikatakan dalam kondisi Deadlock.

- Resource yang bisa dipakai bersama
Deadlock yang terjadi karena permintaan suatu resource yang sedang dipakai oleh proses lain yang sedang menunggu juga. Contoh bisa dilihat pada Gambar 5

c. Kondisi Deadlock
Untuk terjadinya Deadlock, ada berbagai macam

kondisi yang harus terepenuhi. Yaitu:

- *Mutual Exclusion*

  Sebuah resource tidak bisa dipakai oleh 2 proses dalam waktu yang bersamaan. Misal ada resource R, lalu akan diminta oleh 2 proses P1 dan P2. Maka syarat mutual Exclusion akan terpenuhi

- *Hold and Wait*

  Sebuah Proses meminta resource lain namun proses tersebut masih menahan resource awal. Misal dalam kondisi awal proses P sudah memiliki resource M1, namun saat eksekusi, P meminta resource lebih. Maka syarat hold and wait akan terpenuhi

- *No preemption*

  Sebuah proses hanya bisa melepaskan resource miliknya ketika proses itu sudah selesai eksekusi. Jika proses tersebut menjalankan peraturan itu, maka syarat No Preemption akan terpenuhi

- *Circular Wait*

  Dua atau lebih proses yang menunggu 1 sama lain sehingga membentuk sirkuit. Ketika hal itu terjadi, maka syarat Circular Wait akan terpenuhi

Agar Deadlock terjadi, keempat syarat tersebut harus terpenuhi.

d. Deadlock Handling

Cara mengatasi Deadlock ada 3 yaitu

- Deadlock Prevention

  Memastikan salah satu syarat terjadinya deadlock untuk tidak terpenuhi. Dengan begitu kita bisa menghilangkan kemungkinan terjadinya Deadlock.

- Deadlock Detection and Recovery

  Secara Periodik, melakukan pengecekan terhadap Resource-Allocation Graph. Dengan begitu jika terjadi deadlock, kita bisa dengan cepat melakukan sebuah tindakan untuk keluar dari kondisi deadlock tersebut. Ada beberapa cara untuk keluar dari kondisi Deadlock, yaitu

  - ♦ Menggagalkan semua proses yang terlibat dalam kondisi Deadlock
  - ♦ Mengagalkan salah 1 proses hingga tidak terjadi siklus Deadlock

- Deadlock avoidance

  Dengan membuat aturan untuk sebuah Proses agar memberi informasi tentang semua resource apa saja yang mungkin dipakai oleh proses tersebut dari awal pembuatan maupun saat waktu eksekusi nantinya. Sehingga Management Resource akan mengalokasikan resource tersebut kedalam proses dalam statu aman (tidak mungkin terjadi kondisi Deadlock)

e. Deadlock Prevention

Pencegahan Deadlock dengan memastikan salah satu syarat deadlock tidak terpenuhi

- No Mutual Exclusion

  Membatasi hak akses proses pada resource yang dapat dipakai semua proses. Contohnya hanya bisa mengambil nilai dari resource tersebut namun tidak bisa menulis kedalamnya

- No Hold and Wait

  Memberi peraturan pada proses agar tidak meminta reseource jika ia masih memiliki resource

- Allow Preemption

  Ketika sebuah proses P1 meminta resource yang dimiliki oleh proses lain P2, maka semua resource yang proses P1 miliki harus dibebaskan terlebih dahulu. Lalu jika resource yang ia miliki sebelumnya dan resource yang diminta oleh proses P1 sudah bisa dialokasikan, maka akan diberikan pada proses 1

- No Circluar Wait

  Menghitung total resource yang dibutuhkan oleh rantai deadlock proses tersebut, lalu mengeksekusi proses tersebut satu per satu.

f. Deadlock Detection

Dari Proses yang dijalankan pada suatu sistem, dibuat graf yang menggambarkan keterhubungan dengan proses lain dan resource (Resource-Allocating graph). Dalam Graf tersebut ada beberapa kesimpulan dalam penentuan kemungkinan terjadinya Deadlock

- Graf memiliki sirkuit

  Jika Graf memiliki sirkuit, maka dilihat terlebih dahulu tipe resource nya. Jika didalam resource ada banyak stok, maka kondisi deadlock mungkin terjadi. Berarti deadlock juga bisa tidak terjadi. Untuk resource yang hanya memiliki satu stock, maka sudah dipastikan terjadi kondisi deadlock dalam sistem tersebut.

  Dari dua kemungkinan tersebut, Deadlock recovery tetap dilakukan agar kita bisa memastikan sistem tidak akan terdapat kondisi Deadlock nantinya.

- Graf Tidak Memiliki sirkuit

  Jika graf tidak memiliki sirkuit, maka bisa disimpukan bahwa sistem tersebut tidak mungkin terjadi kondisi deadlock (status aman)

g. Deadlock Avoidance

Deadlock avoidance memiliki tujuan untuk

memberikan atau mengalokasikan resource kepada suatu proses dengan kepastian bahwa tidak akan terjadi deadlock di masa depan. Hal tersebut bisa dilakukan dengan cara Memberikan aturan kepada proses untuk menginformasikan daftar dan jumlah total kemungkinan resource yang akan dipakai oleh proses tersebut. Dengan begitu Resource management akan menentukan bahwa alokasi itu akan diberikan atau tidak.

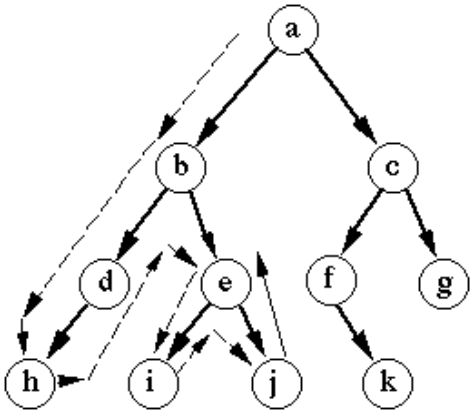
Snapshot at time T_0:

	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

Gambar 7. Contoh daftar kemungkinan total resource pada circular process
Gambar diambil dari http://kuliah.informatika.org/ diakses pada tanggal 5 Mei 2015 jam 07.25

Gambar 7 adalah salah 1 contoh dalam mendaftarkan resource yang diperlukan beserta jumlahnya. Dari daftar tersebut akan dicari jalur pengeksekusian proses sehingga bisa dipastikan tidak ada kemungkinan deadlock di masa depan. Jika tidak ada jalur solusi, maka resource management tidak akan mengalokasikan resource pada proses-proses tersebut.

5. Depth-First-Search
   Depth-First-Search adalah algoritma pencarian solusi pada sebuah pohon ruang status. Sesuai namanya, maka urutan pencarian solusi dalam pohon ruang status dilakukan hingga mencapai ujung terdalam, lalu jika belum mendapat solusi, maka akan dicari ke cabang lain yang terdekat.



Depth-first search

Gambar 8. Urutan pencarian solusi dalam pohon statu dengan menggunakan DFS
Gambar diambil dari http://www.cse.unsw.edu.au/~billw/Justsearch.html diakses pada tanggal 5 Mei 2015 jam 07.35

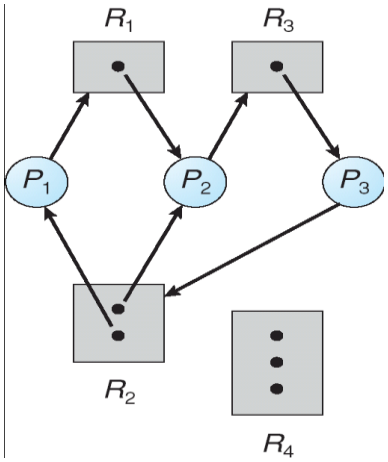
Berikut adalah Pseudocode dari algoritma Depth-First-Search

```
procedure DFS(input v:integer)
{Mengunjungi seluruh simpul graf dengan algoritma pencarian DFS

Masukan: v adalah simpul awal kunjungan
Keluaran: semua simpulyang dikunjungi ditulis ke layar
}
Deklarasi
   w : integer

Algoritma:
   write(v)
   dikunjungi[v]←true
   for w←1 to n do
      if A[v,w]=1 then {simpul v dan simpul w bertetangga }
         if not dikunjungi[w] then
            DFS(w)
         endif
      endif
   endfor
```

Gambar 8. Urutan pencarian solusi dalam pohon statu dengan menggunakan DFS
Gambar diambil dari http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2014-2015/stima14-15.htm diakses pada tanggal 5 Mei 2015 jam 07.38

III. ANALISIS

Seperti yang sudah dijelaskan pada BAB II mengenai mengatasi Deadlock. Salah satu caranya adalah dengan tindakan Mendeteksi dan Memperbaiki. Cara untuk mendeteksi kemungkinan adanya deadlock adalah dengan cara melihat ada atau tidaknya siklus dalam Resource-Allocation Graph. Jika terdapat Siklus dalam graf maupun tipe resourcenya adalah yang memiliki 1 stok atau banyak stok, maka akan selalu ada kemungkinan bahwa 2 kasus ini akan menyebabkan deadlock yang akan berakibat menurunnya utilitas sistem kita karena penggunaan resource yang digunakan proses yang terkena deadlock tidak akan dikembalikan.



Gambar 8. Urutan pencarian solusi dalam pohon statu dengan menggunakan DFS
Gambar diambil dari http://kuliah.informatika.org/ diakses pada tanggal 5 Mei 2015 jam 08.20

Maka untuk mendeteksinya ada atau tidak adanya sirkuit pada Resource-Allocation graph, akan digunakan Algoritma Depth-First-Search. Sengaja digunakan algoritma dasar seperti Depth-First-Search, dan bukan algoritma lanjutan Depth-First-Search sendiri seperti Depth-Limited-Search (DLS) atau Iterative Deepening Search (IDS) karena dalam masalah ini, memang diinginkan pencarian status hingga didapatkan salah satu siklus atau semua node sudah terjangkau. Ketika hasil kembali adalah semua node sudah terjangku, maka kesimpulannya tidak ada siklus dalam Resource-Allocation Graph. Namun jika ditengah algoritma sudah keluar dari proses *loop*, maka kesimpulannya terdapat sirkuit dalam Resource-Allocation Graph dan dicatat status yang menyebabkan siklus tersebut. Lalu pada Akhirnya akan dilakukan perbaikan pada sirkuit tersebut sehingga tidak ada lagi sirkuit didalam graf.

Untuk Implementasinya bisa dengan cara berikut

1. dibuat sebuah array of Boolean yang menyatakan bahwa sebuah simpul dengan indeks yang sama sudah pernah dijamah atau belum.
2. Pilih sebuah simpul , bebas.
3. Lalu lakukan Algoritma Depth-First-Search dengan mencatat semua simpul yang dijamah.
4. Didalam algoritma tersebut, dilakukan sebuah modifikasi. Yaitu untuk setiap simpul. yang diperpanjang, maka akan dilakukan pengecekan pada setiap simpul anaknya. Jika terdapat simpul anak yang sudah pernah dijamah, maka simpul tersebut yang akan diteruskan dan algoritma pencarian berhenti.
5. Jika tidak terdapat simpul anak yang pernah dijamah, lanjutkan algoritma Depth-First-Search
6. Jika sampai akhir algoritma tidak didapat simpul anak yang pernah dijamah, maka graf dikatakan tidak memiliki sirkuit.
7. Jika Ditemukan simpul anak yang pernah dijamah, makan alur program akan keluar dari algoritma pencarian dan dari status simpul yang pernah dijamah, dilakukan penyaringan, agar didapat string atau array yang menyatakan lintasan sirkuit tersebut.
8. Algoritma untuk penyaringannya, bisa berupa pencocokan elemen terakhir dengan elemen sebelumnya yang elemennya sama dan pertama kali ditemukan.
9. Setelah didapat lintasan yang menyebabkan siklus, maka akan dikembalikan kepada sistem operasi untuk diperbaiki.

IV. KESIMPULAN

Untuk mencari siklus yang menyebabkan Deadlock pada sistem, bisa digunakan algoritma dasar seperti Depth-First-Search yang ternyata sangat ampuh,

Algoritma sedasar apapun terkadang bisa sangat berguna. Dengan algoritma dasar seperti Depth-First-Search bisa mengatasi masalah yang dianggap cukup penting. Dengan adanya deadlock, sistem yang kita miliki, bisa menurun. Untuk itu, penting bagi kita untuk belajar sesuatu dari dasar dan memahami konsep dengan baik.

REFERENCES

1. Silberschatz, Abraham; Cagne, Greg; Galvin, Peter Baer (2004). "Chapter 4 - Processes". Operating system concepts with Java (Sixth ed.). John Wiley & Sons. ISBN 0-471-48905-0.
2. http://aliansyah.blog.upi.edu/2013/09/29/model-status-proses-pada-sistem-operasi/
3. https://github.com/angrave/SystemProgramming/wiki/Deadlock,-Part-1:-Resource-Allocation-Graph
4. Silberschatz, Abraham; Cagne, Greg; Galvin, Peter Baer (2004). Operating system concepts (Ninth ed.).
5. https://www.seas.upenn.edu/~cit595/cit595s10/lectures/Deadlocks.pdf

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 29 April 2010

Yoga Adrian Saputra , 13513030