Jurnal Mahajana Informasi, Vol.2 No 2, 2017
e-ISSN: 2527-8290

SIMULASI ALGORITMA BANKER PADA SISTEM ANTRIAN

Markus Bangun
Program Studi Sistem Informasi Universitas Sari Mutiara Indonesia
markusbangun@gmail.com

ABSTRAK

Algoritma Banker dikemukakan oleh Edsger W.Dijkstra dan merupakan salah satu metode untuk menghindari *deadlock*. Algoritma ini disebut algoritma Banker karena memodelkan sebuah bank di kota kecil yang berurusan dengan sekumpulan nasabah yang memohon kredit. Analogi dari algoritma Banker dengan sistem operasi adalah, nasabah merupakan proses-proses yang sedang berjalan, uang (dana yang dimiliki bank) merupakan sumber daya, dan bankir merupakan sistem operasi. Setiap nasabah memiliki batas kredit. Apabila seorang nasabah telah mencapai batas kredit pinjaman, maka diasumsikan nasabah tersebut telah menyelesaikan semua permasalahan bisnisnya dan dapat mengembalikan semua pinjamannya kepada bank. Setiap nasabah dapat memohon kredit pada suatu waktu dan bankir dapat menyetujui atau menolak permohonan tersebut. Jika ditolak, nasabah masih menggenggam dana yang telah dipinjamkan untuknya dan menunggu selama waktu berhingga sampai permohonannya dapat disetujui. Suatu proses disebut *deadlock* jika proses menunggu satu kejadian tertentu yang tidak akan pernah terjadi. Sekumpulan proses berkondisi *deadlock* bila setiap proses yang ada di kumpulan itu menunggu suatu kejadian yang hanya dapat dilakukan proses lain yang juga berada di kumpulan itu. *Deadlock* terjadi ketika proses-proses mengakses secara ekslusif sumber daya, sedang menggenggam sumber daya dan meminta sumber daya lain (yang sedang dipegang oleh proses lain). Salah satu metode untuk mencegah deadlock adalah algoritma Banker. Setelah menyelesaikan perangkat lunak simulasi algoritma Banker ini, perangkat lunak juga menjelaskan dan menampilkan hasil analisis transaksi dengan menggunakan algoritma Banker, sehingga dapat membantu pemahaman terhadap cara kerja algoritma *Banker* dalam mata kuliah sistem operasi. Perangkat lunak simulasi ini merupakan ilustrasi dari proses-proses yang berkompetisi untuk meminta sumber daya kepada sistem operasi.

Kata Kunci : Algoritma Banker, *deadlock*.

I. PENDAHULUAN

1.1 Latar Belakang

Suatu proses disebut *deadlock* jika proses menunggu satu kejadian tertentu yang tidak akan pernah terjadi. Sekumpulan proses berkondisi *deadlock* bila setiap proses yang ada di kumpulan itu menunggu suatu kejadian yang hanya dapat dilakukan proses lain yang juga berada di kumpulan itu. *Deadlock* terjadi ketika proses-proses mengakses secara ekslusif sumber daya, sedang menggenggam sumber daya dan meminta sumber daya lain (yang sedang dipegang oleh proses lain). Salah satu metode untuk mencegah deadlock adalah algoritma Banker.

Algoritma Banker dikemukakan oleh Edsger W.Dijkstra dan merupakan salah satu metode untuk menghindari *deadlock*. Algoritma ini disebut algoritma Banker karena memodelkan

sebuah bank di kota kecil yang berurusan dengan sekumpulan nasabah yang memohon kredit. Analogi dari algoritma Banker dengan sistem operasi adalah, nasabah merupakan proses-proses yang sedang berjalan, uang (dana yang dimiliki bank) merupakan sumber daya, dan bankir merupakan sistem operasi. Setiap nasabah memiliki batas kredit. Apabila seorang nasabah telah mencapai batas kredit pinjaman, maka diasumsikan nasabah tersebut telah menyelesaikan semua permasalahan bisnisnya dan dapat mengembalikan semua pinjamannya kepada bank. Setiap nasabah dapat memohon kredit pada suatu waktu dan bankir dapat menyetujui atau menolak permohonan tersebut. Jika ditolak, nasabah masih menggenggam dana yang telah dipinjamkan untuknya dan menunggu selama waktu berhingga sampai permohonannya dapat disetujui. Bankir hanya memberikan permintaan yang menghasilkan *state* selamat. Permohonan kredit yang akan menghasilkan *state* tidak selamat secara berulang ditolak sampai permohonan tersebut dapat dipenuhi. Persetujuan atau penolakan permohonan kredit ditentukan dengan menggunakan algoritma Safety dan algoritma Resource Request.

1.2 Perumusan Masalah

Uraian masalah dari penelitian ini adalah:

1. bagaimana menerapkan algoritma Banker yang terdiri dari algoritma *Safety* dan algoritma *Resource Request* untuk menentukan keputusan apakah sumber daya dipinjamkan atau tidak.
2. bagaimana membuat atau menggambar objek-objek simulasi dan merancang animasi pergerakan objek dalam sebuah perangkat lunak simulasi.

1.3 Batasan Masalah

Adapun batasan-batasan yang diberikan agar penyelesaian masalah tidak terlalu kompleks adalah sebagai berikut:

1. *Input* dari perangkat lunak adalah,
   a. Jumlah tipe *resource* minimal 1 dan maksimal 5.
   b. *Available*, menunjukkan *resource-resource* yang tersedia untuk setiap tipe, memiliki batasan nilai minimal 1 dan maksimal 100.
   c. *Maximum*, mendefinisikan jumlah *resource* maksimum yang dialokasikan untuk setiap nasabah, memiliki batasan nilai minimal 1 dan maksimal 30.
2. Kondisi simulasi (kondisi awal, permohonan *resources*, besarnya *resources* dan pengembalian *resources*) dapat di-*input* oleh *user* atau dihasilkan secara acak (*random*) oleh komputer.
3. Jumlah tempat pelayanan nasabah (*counter*) dibatasi sebanyak 5 buah.
4. Perangkat lunak akan menampilkan laporan mengenai data-data nasabah yang melakukan transaksi.
5. Laporan tersebut dapat disimpan ke dalam bentuk *text file*.
6. Kecepatan animasi dapat diatur secara manual.

7. Waktu atau lamanya proses simulasi dapat di-*input* oleh *user* dan dibatasi maksimal 3 digit.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah untuk merancang suatu perangkat lunak simulasi algoritma Banker.

1.5 Manfaat Penelitian

Manfaat dari penyusunan penelitian ini, yaitu :

1. Untuk membantu pemahaman mengenai algoritma Banker.
2. Perangkat lunak juga dapat digunakan sebagai fasilitas pendukung dalam proses belajar mengajar, terutama mengenai mata kuliah Sistem Operasi.

II. LANDASAN TEORI

2.1 Sistem Operasi

Secara umum, sebuah sistem komputer terbagi atas *hardware*, sistem operasi, program aplikasi, dan *user*. *Hardware* terdiri atas CPU, memori dan I/O *device* yang merupakan *resources* dasar. Program aplikasi berisi *compiler*, basis data, *games* dan program-program bisnis, yang merupakan suatu cara atau alat dimana *resource-resource* akan diakses untuk menyelesaikan masalah *user*.

Ada beberapa definisi yang dapat diberikan untuk sistem operasi, antara lain:

a. *Software* yang mengontrol *hardware*, hanya berupa program biasa (seperti beberapa *file* pada DOS).
b. Program yang menjadikan *hardware* lebih mudah untuk digunakan.
c. Kumpulan program yang mengatur kerja *hardware* (seperti: mengatur memori, *printer* dll).
d. *Resource Manager* atau *resource allocator* (seperti: mengatur memori, printer dll).
e. Sebagai program pengontrol (program yang digunakan untuk mengontrol program yang lainnya).
f. Sebagai Kernel, yaitu program yang terus-menerus *running* selama komputer dihidupkan.
g. Sebagai *guardian*, yaitu mengatur atau menjaga komputer dari berbagai kejahatan komputer.

2.1.1 Sejarah Perkembangan Sistem Operasi

Sistem operasi telah berevolusi sejak komputer diciptakan. Perkembangan sistem komputer dibagi menjadi empat kurun waktu (generasi). Perkembangan sistem komputer melibatkan perkembangan perangkat keras dan sistem perangkat lunak.

a. Generasi Pertama (1945 – 1955)

Merupakan awal pengembangan sistem komputasi elektronik, mengganti gagasan-gagasan mesin komputasi mekanis. Manusia memerlukan perangkat komputasi untuk memperluas kemampuannya.

Manusia mempunyai batasan komputasi yaitu :

1. Kecepatan penghitung manusia terbatas.
2. Manusia sangat mudah membuat kesalahan.

3. Komputer mekanis mempunyai dua penyebab kelemahan yaitu,
4. Kecepatan komputasi dibatasi inersia bagian-bagian yang bergerak.
5. Transmisi informasi alat-alat mekanis tidak praktis, susah dipakai serta tak handal.

Pada generasi ini belum ada sistem operasi, sistem komputer diberi instruksi yang harus dikerjakan secara langsung.

b. Generasi Kedua (1955 – 1965)

Komputer masa ini adalah *batch processsing system*. *Job* dikumpulkan dalam satu rangkaian kemudian dieksekusi secara berurutan. Pada awal kurun ini, sistem komputer belum dilengkapi sistem operasi, tapi beberapa fungsi dasar sistem operasi telah ada, misalnya FMS (*Fortran Monitoring System*) dan IBSYS. Keduanya merupakan bagian yang fungsinya merupakan komponen sistem operasi. Pada 1964, IBM mengeluarkan keluarga komputer *System*/360. Beragam kelas komputer S/360 dirancang agar kompatibel secara perangkat keras. S/360 menggunakan sistem operasi OS/360. *System* 360 berevolusi menjadi *System* 370.

c. Generasi Ketiga (1965 – 1980)

Perkembangan sistem operasi berlanjut, dikembangkan untuk melayani banyak pemakai interaktif sekaligus. Pemakai-pemakai interaktif berkomunikasi dengan komputer lewat terminal secara *online* (secara langsung dihubungkan) ke komputer. Sistem komputer menjadi :

1. *Multiuser*, yaitu digunakan banyak orang sekaligus.
2. *Multiprogramming*, yaitu melayani banyak program sekaligus.

Multiprogramming berarti komputer melayani banyak proses / *job* (program yang dijalankan) sekaligus pada satu waktu. Cara yang dilakukan untuk ini adalah mempartisi memori menjadi beberapa bagian, dengan satu bagian memori adalah satu *job* berbeda. Saat satu *job* menunggu operasi masukan / keluaran selesai, *job* lain dapat menggunakan pemroses. Teknik ini meningkatkan efisiensi pemroses. Teknik ini memerlukan perangkat keras khusus untuk mencegah satu *job* mengganggu *job* lain.

Karena pemakai-pemakai berinteraksi dengan komputer, komputer harus menanggapi permintaan-permintaan pemakai secara cepat, atau akan menyebabkan produktivitas pemakai menurun drastis. Untuk kebutuhan itu dikembangkan *timesharing*.

Timesharing merupakan varian dari *multiprogramming*, dimana tiap pemakai mempunyai satu terminal *on-line* dengan pemroses hanya memberi layanan pada pemakai yang aktif secara bergantian secara cepat. Pemakai-pemakai akan merasa dilayani terus-menerus, padahal sebenarnya digilir per satuan waktu yang singkat. Karena sumber daya digunakan bersama sering menimbulkan *bottleneck*, maka dikembangkan *spooling*.

Spooling membuat *peripheral* seolah-olah dapat digunakan bersama-sama sekaligus, dapat diakses secara simultan, yaitu dengan cara menyediakan beberapa partisi memori. Saat terdapat permintaan layanan *peripheral*, langsung diterima dan data disimpan lebih dulu di memori yang disediakan (berupa antrian), kemudian dijadwalkan agar secara nyata dilayani oleh *peripheral*.

Usaha pengembangan *timesharing* yang penting adalah sistem CTSS (di MIT), sistem TSS (oleh IBM), Multics, CP/CMS. Setelah itu muncul sistem operasi yang lebih diterima secara umum yaitu UNIX.

d. Generasi Keempat (1980 – 199x)

Sistem operasi tidak lagi hanya untuk satu mode, tapi banyak mode, yaitu mendukung *batch processsing*, *timesharing* dan (*soft*) *real-time applications*.

Generasi ini ditandai dengan berkembang dan meningkatnya kemampuan komputer *desktop* (komputer pribadi) dan teknologi jaringan. Jaringan TCP/IP telah mulai digunakan secara luas oleh kalangan militer, peneliti dan perguruan tinggi.

1. *Network Operating System*

Sistem operasi diperuntukkan bagi jaringan komputer dimana pemakai menyadari keberadaan komputer-komputer yang terhubung. Pada generasi ini juga telah dituntut kenyamanan dalam mengoperasikan sistem komputer. Komputer yang makin ampuh telah sanggup untuk memberi antarmuka grafis yang nyaman. Komputer pribadi telah dinyamankan dengan GUI (*Graphical User Interface*) yaitu antarmuka komputer yang berbasiskan grafis yang interaktif dan menarik. GUI ini dimulai dengan X Windows hasil penelitian di MIT, kemudian Macintosh, Sun View, kemudian disusul Microsoft Windows dan sebagainya.

Pada tahun 1990 dimulasi era komputasi tersebar (*distributed computing*) dimana komputasi-komputasi tidak lagi terpusat (di satu titik), tapi dipecah menjadi subkomputasi-subkomputasi yang dieksekusi di banyak komputer sehingga tercapai kinerja lebih baik. Aplikasi-aplikasi saat ini telah menghendaki adanya kemampuan jaringan.

2. *Distributed Operating System*

Sistem operasi diperuntukkan bagi jaringan komputer, dimana pemakai tidak perlu menyadari keberadaan komputer-komputer yang terhubung. Pengalokasian kerja sudah secara otomatis dilaksanakan pada sistem operasi. Pemakai memandang jaringan komputer sebagai satu uniprosesor besar, walau sebenarnya terdiri dari banyak prosesor (komputer) yang tersebar.

2.2 Proses

Proses adalah entitas dinamis. Proses berisi instruksi dan data, *program counter* dan semua *register* pemroses, dan *stack* berisi data sementara seperti parameter rutin, alamat pengiriman dan variabel-variabel lokal. Istilah lain untuk proses adalah *task*.

2.2.1 Deskripsi Proses

Proses merupakan konsep pokok di sistem operasi. Konsep ini pertama kali dipakai di sistem operasi Multics tahun 60-an. Tema utama perancangan sistem operasi semuanya berkaitan dengan manajemen proses.

Terdapat beragam definisi proses. Salah satunya adalah program yang sedang dieksekusi. Proses merupakan unit kerja terkecil yang secara individu memiliki sumber daya-sumber daya dan dijadwalkan sistem operasi. Sistem operasi mengelola semua proses di sistem dan mengalokasikan sumber daya ke proses-proses sesuai kebijaksanaan untuk memenuhi sasaran sistem. Abstraksi proses merupakan hal mendasar dalam manajemen program-program kongruen.

Beberapa istilah berkaitan dengan proses, antara lain:

1. *Multiprogramming* (*multitasking*).
2. *Multiprocessing*.
3. *Distributed processing / computing*.

2.2.1.1 Multiprogramming (Multitasking)

Multiprogramming adalah manajemen banyak proses pada satu pemroses. *Multiprocessing* telah digunakan untuk suatu konsep, yaitu komputer dengan banyak pemroses di satu sistem komputer dengan masing-masing pemroses melakukan pemrosesan secara independen.

Saat ini, kebanyakan komputer pribadi, *workstation* adalah sistem pemroses tunggal yang menjalankan sistem operasi *multiprogramming* (*multitasking*) seperti MS-Windows 3.0, MS-Windows NT, OS/2 dan Macintosh *System* 7. Banyak proses dijalankan bersamaan, masing-masing proses mendapat bagian memori dan kendali tersendiri. Sistem operasi mengalih-alihkan pemroses di antara proses-proses tersebut.

Program-program yang dijalankan sebenarnya bersifat:

1. Saling tak bergantung (*independen*).

   Proses terpisah satu dari lainnya dan tidak saling berpengaruh.
2. Satu program pada satu saat (*one program at any instant*).

   Pada satu waktu sesungguhnya hanya satu proses yang dilayani pemroses menggunakan *interleave* bukan *overlap* di antara program-program. Dilakukan *interleave* (saling melanjutkan/bersambung). Dalam pandangan pemakai, proses-proses seolah beroperasi secara bersamaan karena pengalihan proses-proses dilakukan secara cepat. Pemroses (level pemrosesan) mengeksekusi satu proses tiap saat dan secara cepat beralih ke proses-proses lain secara bergiliran. Karena pengalihan dilakukan dengan cepat, maka tidak disadari pemakai sehingga menimbulkan efek paralel semu (*pseudoparallelism*).

1.2.1.2 Multiprocessing

Multiprocessing adalah manajemen banyak proses dalam komputer *multiprocessor* (terdapat banyak pemroses di dalamnya). Dulunya sistem ini hanya terdapat di sistem besar, *mainframe* dan minikomputer. Saat ini komputer *workstation* telah dapat dilengkapi *multiprocessor*. Sistem operasi Microsoft Windows NT, UNIX, Linux menyediakan dukungan *multiprocessing*.

2.2.1.3 Distributed Processing

Distributed Processing adalah manajemen banyak proses yang dieksekusi di banyak sistem komputer yang tersebar (terdistribusi). Trend masa datang adalah menuju komputasi tersebar (*distributed computing*). Banyak riset dan pengembangan sistem operasi tersebar di antaranya AMOEBA, MACH, dan sebagainya.

2.3 Deadlock

Proses disebut *deadlock* jika proses menunggu satu kejadian tertentu yang tak akan pernah terjadi. Sekumpulan proses berkondisi *deadlock* bila setiap proses yang ada di kumpulan itu menunggu suatu kejadian yang hanya dapat dilakukan proses lain yang juga berada di kumpulan itu. Proses menunggu kejadian yang tidak akan pernah terjadi. *Deadlock* terjadi ketika proses-proses mengakses secara ekslusif sumber daya. Semua *deadlock* yang terjadi melibatkan persaingan memperoleh sumber daya ekslusif oleh dua proses atau lebih.

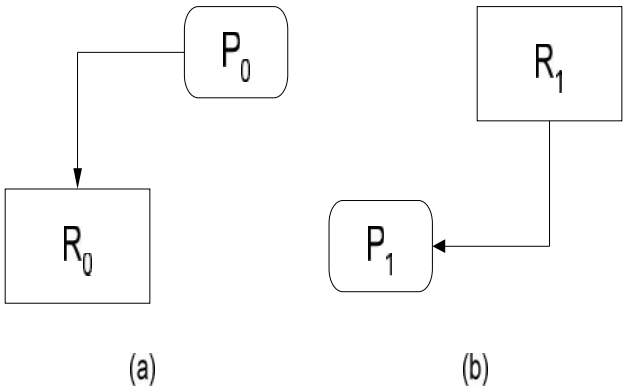
Proses dikatakan sebagai mengalami *starvation* bila proses-proses itu menunggu alokasi sumber daya sampai tak berhingga, sementara proses-proses lain dapat memperoleh alokasi sumber daya. *Starvation* disebabkan bias pada kebijaksanaan atau strategi alokasi sumber daya. Kondisi ini harus dihindari karena tidak adil tetapi dikehendaki penghindaran dilakukan seefisien mungkin.

2.3.1 Model Deadlock

Urutan kejadian pengoperasian perangkat masukan/keluaran adalah:

1. Meminta (request), yaitu meminta pelayanan perangkat masukan / keluaran.
2. Memakai(use), yaitu memakai perangkat masukan/keluaran.
3. Melepaskan(release), yaitu melepaskan pemakaian perangkat masukan / keluaran.

Misalnya, terdapat dua proses P_0 dan P_1 dan dua sumber daya R_0 dan R_1.



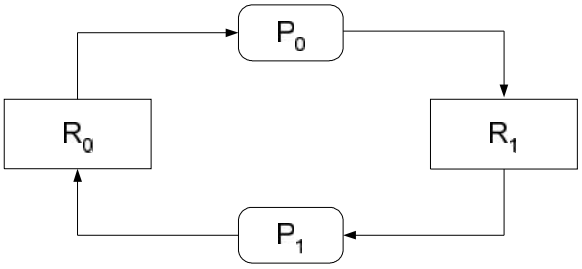
Gambar 2.1 *Graph* meminta sumber daya dan alokasi sumber daya

Gambar 2.1 (a) P_0 meminta sumber daya R_0, ditandai busur (*edge*) berarah dari proses P_0 ke sumber daya R_0. Gambar 2.15 (b) sumber daya R_1 dialokasikan ke P_1, ditandai busur berarah dari

sumber daya R_1 ke proses P_1. Kemudian terjadi skenario berikut,

- P_0 sambil masih menggenggam R_0, meminta R_1.
- P_1 sambil masih menggenggam R_1, meminta R_0.

Kejadian ini mengakibatkan *deadlock* karena sama-sama akan saling menunggu. *Deadlock* tidak hanya terjadi pada dua proses dan dua sumber daya, *deadlock* dpat terjadi dengan melibatkan lebih dari dua proses dan dua sumber daya.

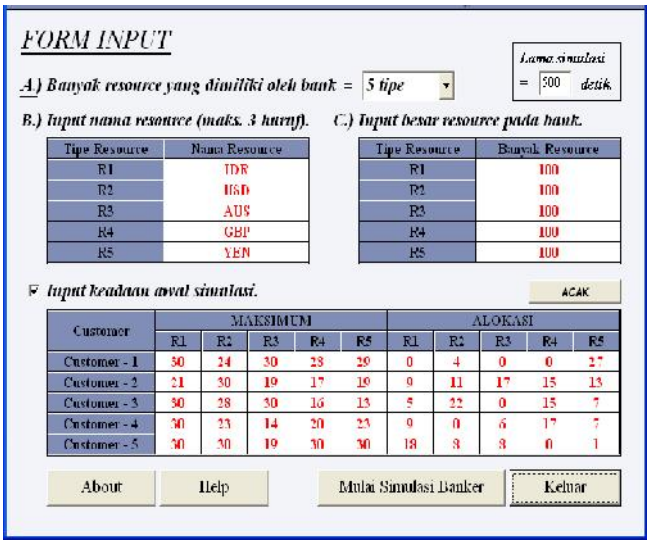


Gambar 2.2 *Graph deadlock* dua proses dan dua sumber daya

III. HASIL DAN PEMBAHASAN

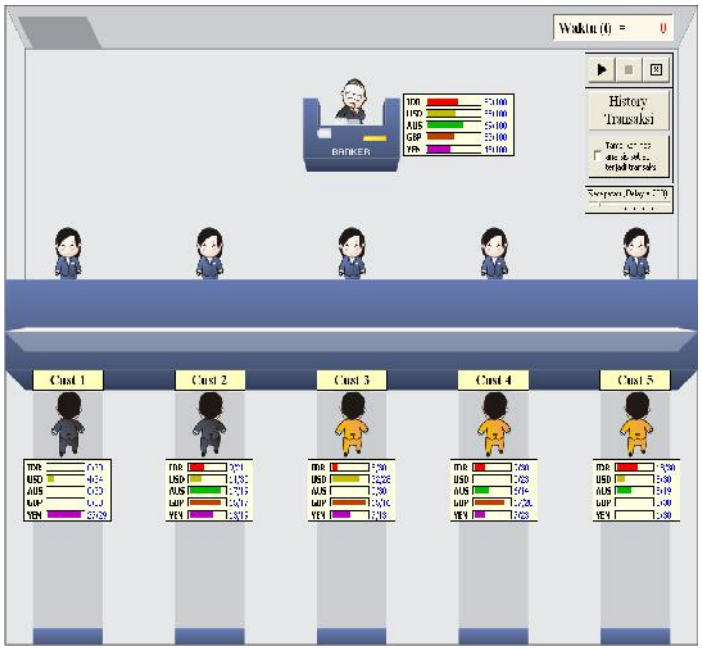
3.1 Pengujian Program

Sebagai contoh untuk menguji *output program*, *input* perangkat lunak (tipe *resource*, nama *resource*, besar *resource* bank dan keadaan awal *customer*) simulasi seperti terlihat pada gambar 3.1 berikut.



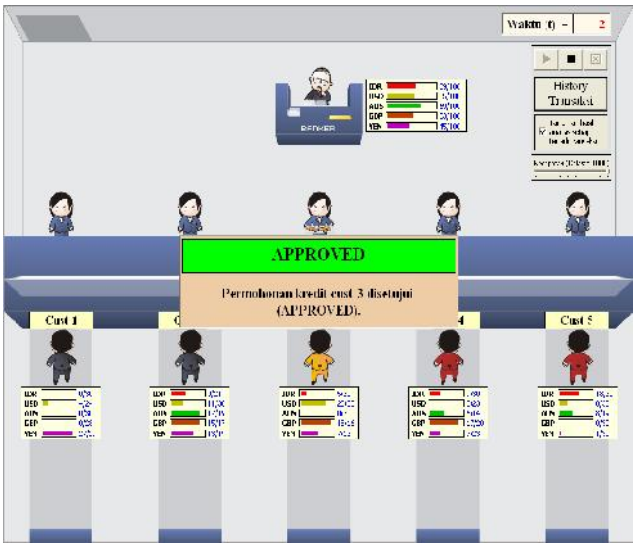
Gambar 3.1 *Input* perangkat lunak

Tampilan awal simulasi seperti terlihat pada gambar 4.2.



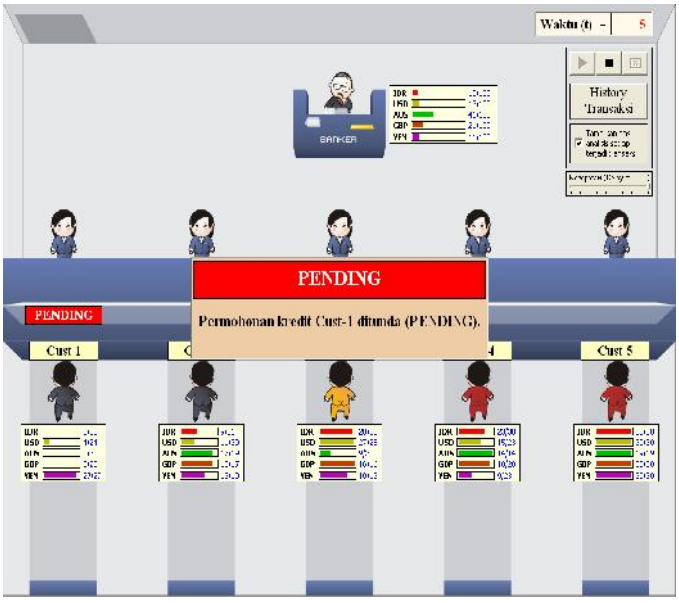
Gambar 3.2 Tampilan awal simulasi

Sebagai contoh, misalkan pada saat t = 2, *customer*-3 memohon kredit (23,5,9,1,3) kepada bankir. Tampilan proses seperti terlihat pada gambar 3.3.



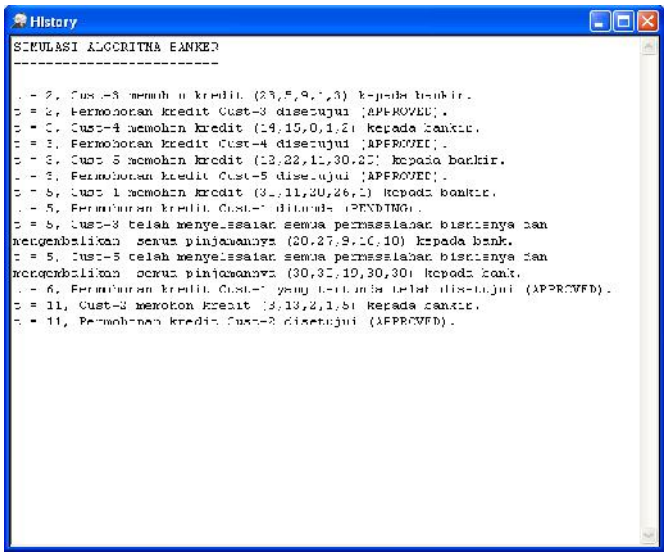
Gambar 3.3 Tampilan simulasi saat *customer*-1 memohon kredit (23,5,9,1,3)

Pada saat *t* = 5, *customer*-1 memohon kredit (30,11,20,26,1) kepada bankir dan permohonan kredit ditunda. Tampilan proses seperti terlihat pada gambar 3.4.



Gambar 3.4 Tampilan simulasi saat *customer*-1 memohon kredit (30,11,20,26,1)

Tampilan *history* perangkat lunak seperti terlihat pada gambar 3.5 berikut.



Gambar 3.5 Tampilan *history* perangkat lunak

IV. KESIMPULAN DAN SARAN

4.1 Kesimpulan

Setelah menyelesaikan perangkat lunak simulasi algoritma Banker ini, penulis menarik kesimpulan sebagai berikut:

1. Perangkat lunak juga menjelaskan dan menampilkan hasil analisis transaksi dengan menggunakan algoritma Banker, sehingga dapat membantu pemahaman terhadap cara kerja algoritma *Banker* dalam mata kuliah sistem operasi.
2. Perangkat lunak simulasi ini merupakan ilustrasi dari proses-proses yang berkompetisi untuk meminta sumber daya kepada sistem operasi.
3. Algoritma Banker adalah suatu metode untuk mencegah *deadlock*. Namun, algoritma Banker tidak pernah diterapkan dalam sistem operasi yang sebenarnya, karena algoritma Banker membatasi jumlah *resource* maksimum untuk

setiap proses, sedangkan proses-proses biasanya belum dapat mengetahui jumlah *resource* maksimum yang dibutuhkannya ketika diciptakan.

4.2 Saran

Penulis ingin memberikan beberapa saran yang mungkin dapat membantu dalam pengembangan perangkat lunak ini yaitu :

1. Proses simulasi dapat ditingkatkan dengan membangun dan menambahkan animasi yang lebh baik dan lebih detil, seperti: animasi kasir menyerahkan uang, animasi *customer* memasuki bank dengan membuka pintu dan animasi lainnya.
2. Perangkat lunak dapat dikembangkan dengan menambahkan jumlah tempat pelayanan yang lebih banyak.
3. Perangkat lunak dapat dikembangkan dengan menambahkan *database* mengenai *customer* untuk transaksi di bank.

DAFTAR PUSTAKA

Hariyanto.B, **Sistem Operasi**, Edisi2, Informatika, Bandung, 1999.

Hariyanto.B, **Sistem Operasi Lanjut**, Informatika, Bandung, 2003.

Kusumadewi.S, **Sistem Operasi**, Edisi2, Graha Ilmu, Jakarta, 2000.

Novian.A, **Panduan MS. Visual Basic 6**, Andi, Yogyakarta, 2004.

Ramadhan.A, **MS. Visual Basic 6 (Seri Penuntun Praktis)**, PT. Elex Media Komputindo, Jakarta, 2004.

Supardi.Y, **Microsoft Visual Basic 6.0 Untuk Segala Tingkat**, PT. Elex Media Komputindo, Jakarta, 2006.

http://www.rsu.edu/faculty/PMacpherson/Programs/banker.html.

http://en.wikipedia.org/wiki/Banker's_algorithm.html.

http://www.algana.co.uk/Algorithms/OperatingSystems/Banker/Banker.htm

http://www.if.uidaho.edu/~bgray/classes/cs341/doc/banker.html.

http://www.buchli.org/frank/work/os/deadlock.html.

.http://www.ilmukomputer.com/umum/ibam/ibam-os-html.zip.