PERTEMUAN 5:

THREAD

A. TUJUAN PEMBELAJARAN

Pada bab ini akan dijelaskan mengenai thread, Anda harus mampu:

1.1 Pengertian thread

1.2 Tread dalam proses

1.3 Penjadwalan Thread

B. URAIAN MATERI

Tujuan Pembelajaran 1.1:
Thread

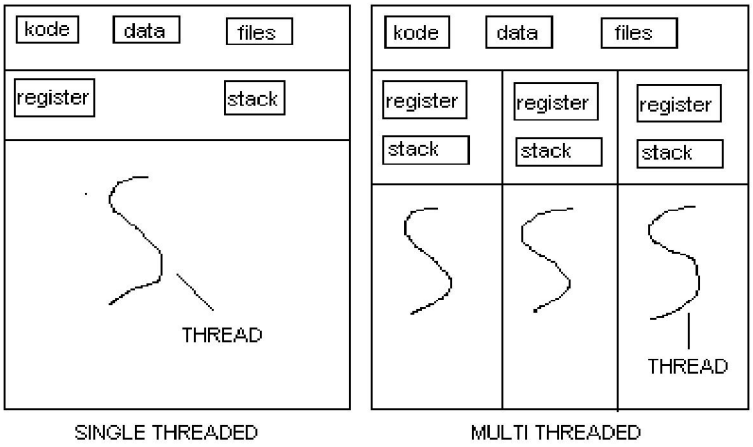
Pengertian Thread

Thread atau yang disebut proses ringan (lightweight) adalah unit dasar dari utilisasi CPU. Di dalamnya terdapat ID thread, program counter, register, dan stack. Dan saling berbagi dengan thread lain dalam proses yang sama. Dengan menggunakan fasilitas thread , CPU dapat secara intensif membagi diantara per thread tanpa menggunakan manajemen memori.

Meskipun context switch pada thread masih memerlukan register set switcj tetapi pembuatan thread lebih murah dibandingkan context switch diantara heavy-weight.

Thread dalam proses

Perbedaan antara proses dengan thread tunggal dengan proses dengan thread yang banyak adalah proses dengan thread yang banyak dapat mengerjakan lebih dari satu tugas pada satu satuan waktu.



Gambar : Proses thread

Keuntungan dan kerugian Multithreading

Multiprocessing merupakan penggunaan dua atau lebih CPU dalam sebuah sistem komputer. Multitasking merupakan metode untuk menjalankan lebih dari satu proses dimana terjadi pembagian sumberdaya seperti CPU. Multithreading adalah cara pengeksekusian yang mengizinkan beberapa thread terjadi dalam sebuah proses, saling berbagi sumber daya tetapi dapat dijalankan secara independen.

Dalam sistem yang menerapkan multithreading memiliki 4 keuntungan,diantara :

1. Responsi

   Membuat aplikasi yang interaktif menjadi multithreading dapat membuat sebuah program terus berjalan meskipun sebagian dari program tersebut diblok atau melakukan operasi yang panjang, karena itu dapat meningkatkan respons kepada pengguna. Sebagai contohnya dalam web browser yang multithreading, sebuah thread dapat melayani permintaan pengguna sementara thread lain berusaha menampilkan image.

2. Berbagi sumber daya

Thread berbagi memori dan sumber daya dengan thread lain yang dimiliki oleh proses yang sama. Keuntungannya adalah mengizinkan sebuah aplikasi untuk mempunyai beberapa thread yang berbeda dalam lokasi memori yang sama.

3. Ekonomi

Dalam pembuatan sebuah proses banyak dibutuhkan pengalokasian memori dan sumber daya. Alternatifnya adalah dengan penggunaan thread, karena thread berbagi memori dan sumber daya proses yang memilikinya maka akan lebih ekonomis untuk membuat dan context switch thread. Akan susah untuk mengukur perbedaan waktu antara proses dan thread dalam hal pembuatan dan pengaturan, tetapi secara umum pembuatan dan pengaturan proses lebih lama dibandingkan thread. Pada Solaris, pembuatan proses lebih lama 30 kali dibandingkan pembuatan thread, dan context switch proses 5 kali lebih lama dibandingkan context switch thread.

4. Utilisasi arsitektur multiprocessor

Keuntungan dari multithreading dapat sangat meningkat pada arsitektur multiprocessor, dimana setiap thread dapat berjalan secara pararel di atas processor yang berbeda. Pada arsitektur processor tunggal, CPU menjalankan setiap thread secara bergantian tetapi hal ini berlangsung sangat cepat sehingga menciptakan ilusi pararel, tetapi pada kenyataannya hanya satu thread yang dijalankan CPU pada satu-satuan waktu (satu-satuan waktu pada CPU biasa disebut time slice atau quantum).

Adapun kerugian yang terjadi dalam menerapkan multithreading, yaitu :

1. Jika digunakan secara berlebihan, multithreading akan berdampak pada pemborosan resource dan CPU yang dialokasikan untuk switching threads. Misalnya jika heavy disk I/O terlibat, akan lebih cepat jika

hanya memiliki 1 atau 2 thread yang melaksanakan tugas secara berurutan, daripada menggunakan multithread yang masing-masing mengeksekusi sebuah task pada waktu yang sama.

2. Sistem yang memiliki kecepatan prosesor dan memory yang cenderung sama, sehingga tidak ada efisiensi yang hilang (mengacu kepada latency), tidak akan memperoleh peningkatan bandwidth yang signifikan jika menggunakan multithreading

3. Multithreading menghasilkan program yang lebih kompleks. Menggunakan multiple thread sendiri tidak akan menciptakan kerumitan, tapi interaksi antar thread-lah yang mengakibatkan kompleksitas tersebut.

4. Thread yang banyak bisa saling berinterferensi ketika saling berbagi sumber daya hardware seperti cache.

Model multithreading

Beberapa terminologi yang akan di bahas , yaitu :

- User (pengguna) thread

  User thread didukung di atas kernel dan diimplementasi oleh thread library pada user level. Library menyediakan fasilitas untuk pembuatan thread, penjadualan thread, dan managemen thread tanpa dukungan dari kernel.

  Adapun kelemahannya yang dialami yaitu apabila kernelnya merupakan thread tunggal maka apabila salah satu user-level thread menjalankan blocking system call maka akan mengakibatkan seluruh proses diblok walau pun ada thread lain yang dapat jalan dalam aplikasi tersebut

- Kernel tread

Kernel thread didukung langsung oleh sistem operasi. Pembuatan, penjadualan, dan managemen thread dilakukan oleh kernel pada kernel space. Pembuatan dan pengaturan kernel thread lebih lambat dibandingkan user thread.

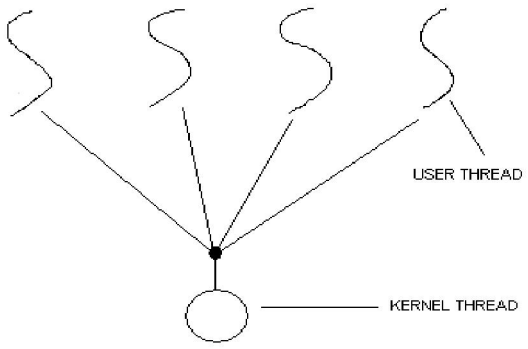
Keuntungannya adalah jika sebuah thread menjalankan blocking system call maka kernel dapat menjadualkan thread lain di aplikasi untuk melakukan eksekusi. Pada lingkungan multiprocessor, kernel dapat menjadual thread-thread pada processor yang berbeda.

Contoh sistem operasi yang mendukung kernel thread adalah Windows NT, Solaris, Digital UNIX.

Multitreading memiliki beberapa model-model,diantarany adalah :

- Many to one model

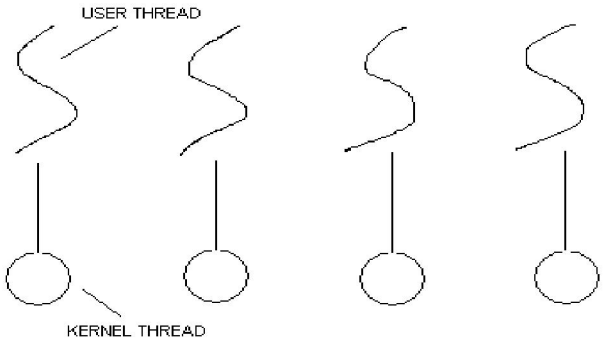
Memetakan banyak user-level thread ke satu kernel thread. Pengaturan thread dilakukan di user space. Efisien tetapi ia mempunyai kelemahan yang sama dengan user thread tidak dapat berjalan secara pararel pada multiprocessor.



Gambar : Many to one model

v One to one model

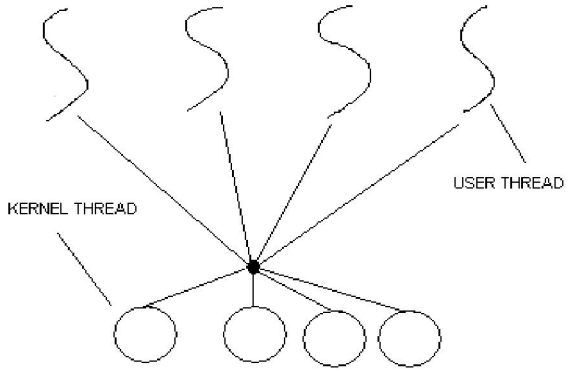
Memetakan setiap user thread ke kernel thread.Menyediakan lebih banyak concurrency dibandingkan Many-to-One model. Keuntungannya sama dengan keuntungan kernel thread. Kelemahannya setiap pembuatan user thread membutuhkan pembuatan kernel thread yang dapat menurunkan performa dari sebuah aplikasi. Sistem operasi yang mendukung One-to-One model adalah Windows NT dan OS/2



Gambar : One to one model

v Many to many model

Multiplexes banyak user-level thread ke kernel thread yang jumlahnya lebih kecil atau sama banyaknya dengan user-level thread. Jumlah kernel thread dapat spesifik untuk sebagian aplikasi atau sebagian mesin. Developer dapat membuat user thread sebanyak yang diperlukan, dan kernel thread yang bersangkutan dapat bejalan secara pararel pada multiprocessor. Ketika suatu thread menjalankan blocking system call maka kernel dapat menjadualkan thread lain untuk melakukan eksekusi.Sistem operasi yang mendukung model ini adalah Solaris, IRIX, dan Digital UNIX.



Gambar : Many to many model

Tread pools

Pada web server yang menerapkan multithreading ada dua masalah yang timbul:

- Ukuran waktu yang diperlukan untuk menciptakan thread yang melayani permintaan yang diajukan pada kenyataannya thread dibuang seketika sesudah ia menyelesaikan tugasnya.
- Pembuatan thread yang tidak terbatas jumlahnya dapat menurunkan performa dari sistem.

Solusinya adalah dengan penggunaan Thread Pools, yaitu sekumpulan thread yang mengantri untuk mengerjakan tugas Cara kerjanya adalah dengan membuat beberapa thread pada proses startup dan menempatkan mereka ke pools, dimana mereka duduk diam dan menunggu untuk bekerja. Jadi, ketika server menerima permintaan, ia akan membangunkan thread dari pool dan jika thread tersedia maka permintaan tersebut akan dilayani. Ketika thread sudah selesai mengerjakan tugasnya maka ia kembali ke pool dan menunggu pekerjaan lainnya. Bila tidak ada thread yang tersedia pada saat dibutuhkan maka server menunggu sampai ada satu thread yang bebas.

Keuntungan thread pool adalah:

a. Biasanya lebih cepat untuk melayani permintaan dengan thread yang ada dibandingkan menunggu thread baru dibuat.

b. Thread pool membatasi jumlah thread yang ada pada suatu waktu. Hal ini penting pada sistem yang tidak dapat mendukung banyak thread yang berjalan secara concurrent. Jumlah thread dalam pool dapat tergantung dari jumlah CPU dalam sistem, jumlah memori fisik, dan jumlah permintaan klien yang concurrent.
c. Pembuatan jumlah thread yang tepat dapat meningkatkan performa serta sistem yang lebih stabil.

Pembatalan Thread

Thread Cancellation ialah pembatalan thread sebelum tugasnya selesai. Misalnya hendak mematikan Java Virtual Machine (JVM) pada program Java. Maka sebelum JVM dimatikan seluruh thread yang berjalan harus dibatalkan terlebih dahulu. Contoh lain adalah pada masalah search. Apabila sebuah thread mencari sesuatu dalam database dan menemukan serta mengembalikan hasilnya, thread sisanya akan dibatalkan. Thread yang akan diberhentikan biasa disebut target thread.
Pemberhentian target Thread dapat dilakukan dengan 2 cara:

a. Asynchronous cancellation. Suatu thread seketika itu juga membatalkan target thread.

b. Deferred cancellation. Suatu thread secara periodik memeriksa apakah ia harus batal, cara ini memperbolehkan target thread untuk membatalkan dirinya secara terurut.

Hal yang sulit dari pembatalan thread ini adalah ketika terjadi situasi dimana sumber daya sudah dialokasikan untuk thread yang akan dibatalkan. Selain itu kesulitan lain adalah ketika thread yang dibatalkan sedang meng-update data yang ia

bagi dengan thread lain. Hal ini akan menjadi masalah yang sulit apabila digunakan asynchronous cancellation. Sistem operasi akan mengambil kembali sumber daya dari thread yang dibatalkan tetapi seringkali sistem operasi tidak mengambil kembali semua sumber daya dari thread yang dibatalkan. Alternatifnya adalah dengan menggunakan deffered cancellation. Cara kerja dari deffered cancellation adalah dengan menggunakan satu thread yang berfungsi sebagai pengindikasi bahwa target thread hendak dibatalkan. Tetapi pembatalan hanya akan terjadi jika target thread memeriksa apakah ia harus batal atau tidak. Hal ini memperbolehkan thread untuk memeriksa apakah ia harus batal pada waktu dimana ia dapat dibatalkan secara aman yang aman. Thread
merujuk sebagai cancellation points.

Pada umumnya sistem operasi memperbolehkan proses atau thread untuk dibatalkan secara asynchronous. Tetapi Pthread API menyediakan deferred cancellation. Hal ini berarti sistem operasi yang mengimplementasikan Pthread API akan mengizinkan deferred cancellation.

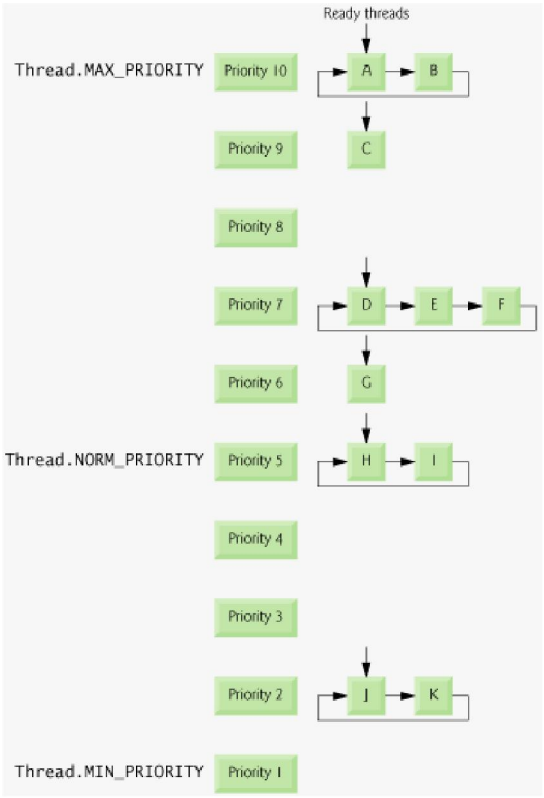
Penjadwalan Thread

Begitu dibuat, thread baru dapat dijalankan dengan berbagai macam penjadwalan. Kebijakan penjadwalanlah yang menentukan setiap proses, di mana proses tersebut akan ditaruh dalam daftar proses sesuai proritasnya dan bagaimana ia bergerak dalam daftar proses tersebut.

Untuk menjadwalkan thread, sistem dengan model multithreading many to many atau many to one menggunakan:

a. Process Contention Scope (PCS). Pustaka thread menjadwalkan thread pengguna untuk berjalan pada LWP (lightweight process) yang tersedia.

b. System Contention Scope (SCS). SCS berfungsi untuk memilih satu dari banyak thread,kemudian menjadwalkannya ke satu thread tertentu (CPU / Kernel).



Gambar : penjadwalan thread

Thread Linux

Ketika pertama kali dikembangkan, Linux tidak didukung dengan threading di dalam kernelnya,tetapi dia mendukung proses-proses sebagai entitas yang dapat dijadwalkan melalui clone() system calls. Sekarang Linux mendukung penduplikasian proses menggunakan system call clone() dan fork(). Clone() mempunyai sifat mirip dengan fork(), kecuali dalam hal pembuatan salinan dari proses yang dipanggil dimana ia membuat sebuah proses yang terpisah yang berbagi address space dengan proses yang dipanggil. Pembagian address space dari parent process memungkinkan cloned task bersifat mirip dengan thread yang terpisah. Pembagian address space ini dimungkinkan karena proses direpresentasikan di dalam Kernel Linux. Didalam Kernel Linux setiap proses direpresentasikan sebagai sebuah struktur data yang unik. Jadi, daripada menciptakan yang baru maka struktur data yang baru mengandung pointer yang menunjuk ke tempat dimana data berada. Jadi ketika fork() dipanggil, proses yang baru akan tercipta beserta duplikasi dari segala isi di struktur data di

parent process, namun ketika clone() dipanggil, ia tidak menduplikasi parent process-nya tetapi menciptakan pointer ke struktur data pada parent process yang memungkinkan child process untuk berbagi memori dan sumber daya dari parent process-nya. Project Linux Thread menggunakan system call ini untuk mensimulasi thread di user space. Sayangnya, pendekatan ini mempunyai beberapa kekurangan, khususnya di area signal handling, scheduling, dan interprocess synchronization primitive.

Untuk meningkatkan kemampuan Thread Linux, dukungan kernel dan penulisan ulang pustaka thread sangat diperlukan. Dua project yang saling bersaing menjawab tantangan ini. Sebuah tim yang terdiri dari pengembang dari IBM membuat NGPT (Next Generation POSIX Threads). Sementara pengembang dari Red Hat membuat NPTL (Native POSIX Thread Library). Sebenarnya Linux tidak membedakan antara proses dan thread. Dalam kenyataannya, Linux lebih menggunakan istilah task dibandingkan proses dan thread ketika merujuk kepada pengaturan alur pengontrolan di dalam program.

Pada sistem operasi lain seperti Mac OS X, terdapat lima thread API yang berbeda, yaitu: Mach thread, POSIX thread (pthreads), Cocoa thread (NSThreads), Carbon MP tasks, dan Carbon Thread Manager. Akan tetapi tidak mudah untuk menentukan suatu thread mengerjakan tugas yang mana, sehingga dibuatlah suatu siklus yang membuat masing-masing thread bekerja secara bergantian.

C. SOAL LATIHAN/TUGAS

1. Jelaskan penegertian thread?
2. Jelaskan pembatalan thread?
3. Jelaskan penjadwalan thread?

D. DAFTAR PUSTAKA

Buku

Bambang Hariyanto. 1997. Sistem Operasi, Bandung:Informatika Bandung.

Dali S. Naga. 1992. Teori dan Soal Sistem Operasi Komputer,Jakarta: Gunadarma.

Operating System Concepts (6th or 7th Edition). Silberschatz, Galvin, Gagne, ISBN: 0-471-25060-0. Wiley

Silberschatz Galvin. 1995. 4 Edition Operating System Concepts: Addison Wesley.

Sri Kusumadewi. 2000. Sistem Operasi. Yogyakarta: J&J Learning.

Tanenbaum, A.1992. Modern Operating Systems.New York: Prentice Hall

Link and Sites:

http://www.ilmukomputer.com
http://vlsm.bebas.org
http://www.wikipedia.com