Tugas 7
Sistem Operasi

Nama : Rheza Dewangga Rendragraha

Kelas: 1 D4 Teknik Informatika B

NRP: 2110191044

1. Berikan tiga contoh pemrograman di mana multithreading memberikan kinerja yang lebih baik daripada solusi single-threaded.
    - Server Web yang melayani setiap permintaan di threads terpisah. Misalnya, server web menerima permintaan klien untuk menampilkan halaman web, gambar, suara dan sebagainya. Server web mungkin memiliki beberapa klien yang mengakses secara bersamaan. Salah satu solusinya adalah menjalankan server sebagai satu proses yang menerima permintaan. Saat permintaan dibuat, server dapat membuat thread yang baru guna melayani permintaan tambahan.
    - Aplikasi paralel seperti perkalian matriks di mana bagian-bagian berbeda dari matriks dapat dikerjakan secara paralel.
    - Program GUI interaktif seperti debugger di mana threads digunakan untuk memantau input user, threads lain mewakili aplikasi yang sedang berjalan, dan threads ketiga memantau kinerja.

2. Apa dua perbedaan antara user-level threads dan kernel-level threads ? Dalam keadaan apa salah satu jenis lebih baik dari yang lain?
    - Pada user-level threads didukung kernel serta diimplemetasikan dengan threads library , sedangkan kernel-level threads dikelola langsung oleh sistem operasi.
    - Pada sistem yang menggunakan pemetaan M : 1 atau M : N, user threads menyediakan fasilitas untuk pembuatan threads, penjadwalan threads, dan manajemen threads tanpa dukungan dari kernel melalui threads library dan kernel (kernel space) menjadwalkan kernel threads.
    - Kernel threads tidak perlu dikaitkan dengan suatu proses di mana setiap user threads adalah bagian dari suatu proses. Kernel threads umumnya lebih mahal untuk dirawat daripada user threads karena harus direpresentasikan dengan struktur data kernel.
    - Pengaturan pada kernel-level threads dilakukan oleh sistem operasi, sehingga pembuatan dan pengaturan kernel-level threads lebih lambat dibandingkan user-level threads.

3. Jelaskan tindakan yang dilakukan oleh kernel untuk beralih konteks antara kernel level threads.
    - Pada saat alih konteks terjadi, kernel akan menyimpan konteks dari proses lama kedalam PCB-nya dan mengisi konteks yang telah disimpan dari proses baru yang telah terjadual untuk berjalan. Pergantian waktu konteks adalah murni overhead, karena sistem ini melakukan pekerjaan yang tidak perlu. Kecepatannya bervariasi dari mesin ke mesin, bergantung pada kecepatan memori, jumlah register yang harus di copy, dan keberadaan instruksi khusus (seperti instruksi tunggal untuk mengisi atau menyimpan seluruh register).
    - Pergantian konteks antara kernel threads biasanya membutuhkan penyimpanan nilai register CPU dari threads yang dialihkan dan mengembalikan register CPU dari threads baru yang dijadwalkan.

4. Sumber daya apa yang digunakan saat threads dibuat? Bagaimana mereka berbeda dari yang digunakan ketika suatu proses dibuat?
    - Karena threads lebih kecil dari suatu proses, pembuatan threads biasanya menggunakan sumber daya yang lebih sedikit daripada pembuatan proses. Membuat suatu proses membutuhkan alokasi blok kontrol proses (PCB), struktur data yang agak besar. PCB mencakup peta memori, daftar file yang terbuka, dan variabel lingkungan. Mengalokasikan dan mengelola peta memori biasanya merupakan aktivitas yang paling memakan waktu. Menciptakan user atau kernel threads melibatkan pengalokasian struktur data kecil untuk menampung set register, stack, dan prioritas.

5. Asumsikan bahwa sistem operasi memetakan thread tingkat pengguna ke kernel menggunakan model many-to-many dan pemetaan dilakukan melalui LWP. Selain itu, sistem ini memungkinkan pengembang untuk membuat threads real-time untuk digunakan dalam sistem real-time. Apakah perlu untuk mengikat threads real-time ke LWP? Jelaskan.
    - Iya. Pengaturan waktu sangat penting untuk aplikasi real-time. Jika threads ditandai sebagai real-time tetapi tidak terikat pada LWP, threads mungkin harus menunggu untuk dilampirkan ke LWP sebelum dijalankan. Pertimbangkan jika threads realtime sedang berjalan (dilampirkan ke LWP) dan kemudian dilanjutkan ke blok (yaitu harus melakukan I / O, telah didahului oleh threads real-time yang memiliki prioritas tinggi, dll.) Saat threads real-time diblokir, LWP yang dilampirkan telah ditugaskan ke threads lain. Ketika threads real-time telah dijadwalkan untuk berjalan kembali, threads tersebut harus menunggu terlebih dahulu untuk dilampirkan ke LWP. Dengan mengikat LWP ke threads real-time, Anda memastikan threads akan dapat berjalan dengan penundaan minimal setelah dijadwalkan.