PENGENALAN KONTROL INPUT/OUTPUT

Pengaksesan (membaca dan menulis) data pada alat penyimpanan sekunder memerlukan banyak aktifitas pada programmer aplikasi. Pemrograman bahasa komputer memungkinkan programmer untuk menjabarkan teknik organisasi berkas yang agak kompleks (missal pada organisasi berkas indeks-sequential) dengan pernyataan bahasa yang sangat sederhana. Dukungan sistem manajemen berkas mengubah pernyataan-pernyataan tersebut menjadi instruksi input/output tingkat bawah (low-level). Demikian pula sebuah permintaan sederhana dari programmer terhadap READ atau WRITE sebuah record pada berkas, secara khusus diperlukan suatu urutan yang komplek dari peralatan dukungan operasi manajemen. Bagian-bagian ini biasanya tidak dapat dilihat oleh programmer. Pekerjaan programmer akan menjadi lebih sukar jika pekerjaan itu menyangkut operasi control input/output yang lebih terinci. Sebuah sistem control I/O bertujuan untuk memberikan bantuan kepada user untuk memungkinkan mereka mengakses berkas, tanpa memperhatikan detail dari karakteristik dan waktu penyimpanan.

Kontrol I/O menyangkut manajemen berkas dan peralatan manajemen yang merupakan bagian dari sistem operasi. Bagian lain dari sistem operasi mencakup sebuah pengaturan memori primer (main memory) dan sebuah pengatur proses dan penjadwalan. Kadang-kadang bagian dari sistem operasi yang melaksanakan control I/O ini disebut pengawas input/output (supervisor I/O). Akses berkas memerlukan dukungan manajemen berkas, yang memberikan teknik organisasi berkas dan dukungan alat manajemen, yang memberikan akses ke alat penyimpan fisik. Pada beberapa sistem operasi sebuah pengatur berkas (file manager) dan sebuah pengatur alat (device manager) berbeda satu sama lain, sedang pada sistem lain, mereka digabungkan untuk membentuk fungsi control I/O tunggal. Tugas dari sistem kontrol I/O sangat banyak dan beraneka ragam.

Beberapa tugas yang dikerjakan oleh kontrol I/O adalah sbb:

1. Memelihara direktori dari berkas dan lokasi informasi.
2. Menentukan jalan (pathway) bagi aliran data antara memori primer (main memory) dan alat penyimpan sekunder.
3. Mengkoordinasi komunikasi antara CPU dan alat penyimpanan sekunder, dan sebaliknya, termasuk:

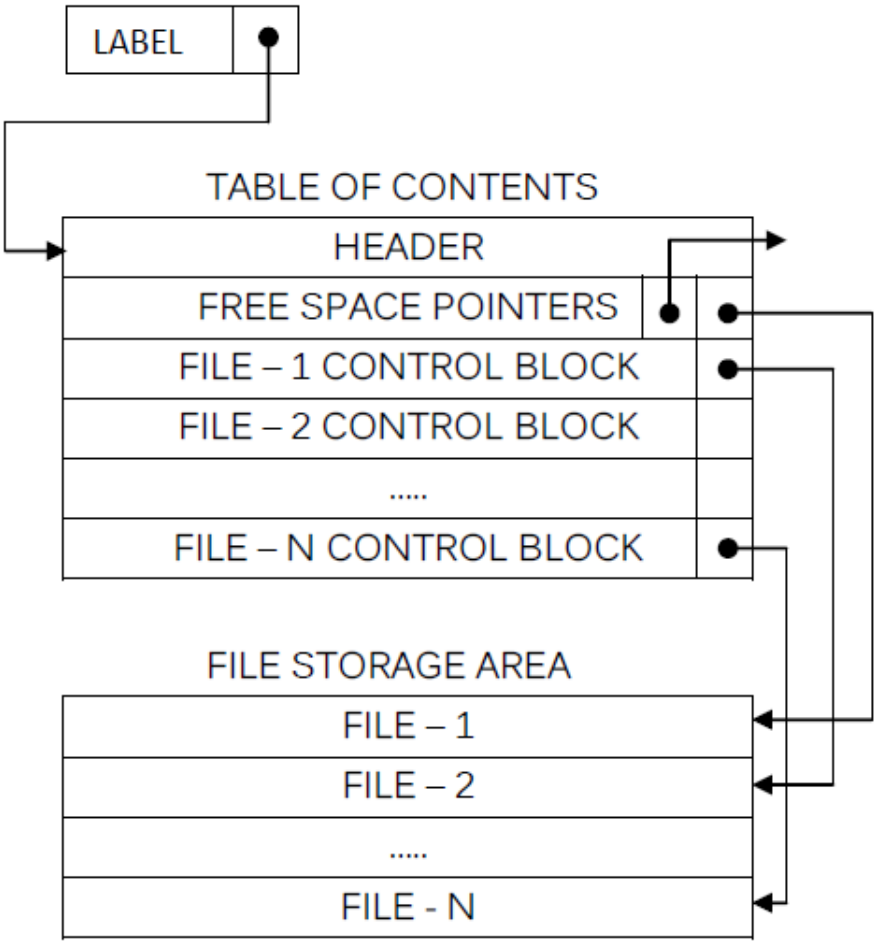
a. Mengatur/menangani ketidakseimbangan kecepatan pengiriman data antara CPU dengan alat penyimpanan sedemikian rupa, sehingga CPU tidak menunggu terlalu lama (membuang waktu) untuk menyelesaikan pekerjaan I/O.
b. Mengatur data sedemikian rupa, sehingga data dapat disimpan, bila pengirim (CPU atau alat penyimpan sekunder) dan penerima (alat penyimpanan sekunder atau CPU) tidak siap pada waktu yang bersamaan.

4. Menyiapkan berkas penggunaan I/O.
5. Mengatur berkas, bila penggunaan input atau output telah selesai.

A. DIREKTORI BERKAS DAN KONTROL INFORMASI

Sebelum berkas dapat di akses oleh sebuah program, sistem operasi harus mengetahui pada alat penyimpanan sekunder yang mana berkas tersebut berada. Hampir semua sistem operasi menggunakan beberapa jenis struktur direktori yang digunakan untuk menyimpan trek dari berkas pada sistem manajemen berkas. Direktori yang diperlihatkan pada gambar dibwah ini adalah untuk satu unit dari penyimpanan sekunder. Labelnya berisi indentifikasi informasi, akses kontrol informasi, dan sebuah primer yang menunjuk ke isi tabel, yang berisi kontrol blok untuk setiap berkas pada unit tersebut.

Sebuah kontrol blok berisi informasi tentang nama dari berkas, atributnya (seperti panjang record, ukuran blok, organisasi berkas), dan batasnya pada media penyimpanan. Sebuah kontrol blok menunjukkan awal dari berkas yang bersangkutan. Jadi bila sebuah berkas dicari, isi tabel dari unit yang dimaksud, diperiksa untuk menemukan berkas pada alat penyimpan.



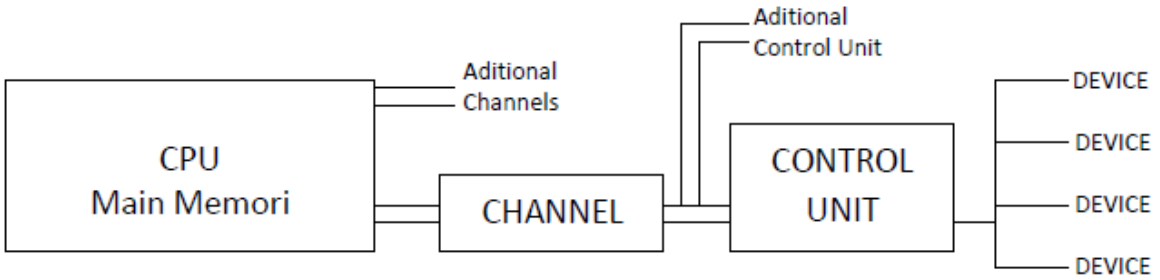
Sistem manajemen berkas yang berbeda, menggunakan struktur yang berbeda untuk blok kontrol berkasnya. Sebuah sistem representative/perwalian membentuk dua tabel untuk menjelaskan setiap berkas. Yang pertama, berisi informasi pada organisasi berkas dan pola pemrosesan, deskripsi ukuran dan jenis record, ukuran pemblokan, kesalahan hitung dan flag, informasi yang digunakan oleh rutin sistem kontrol I/O untuk menentukan status berkas dan posisinya, dan nama dari logika berkas (eksternal). Juga berisi sebuah pointer pada tabel deskriptif berkas kedua, yang dihubungkan dengan sebuah alat penyimpan tertentu dan berisi informasi tentang karakteristik fisik dan status alat yang sedang digunakan.

Alat pengontrol rutin menggunakan tabel ini untuk menetapkan (meng-interface) permintaan terhadap berkas dan untuk memonitor status aktifitas input/output berkas. Kontrol informasi pada sistem ini terpecah antar tabel dengan tujuan untuk memisahkan pendeskripsian berkas dan alat.

B. KONTROL PERALATAN

Aktivitas input/output terutama mencakup perpindahan data antara memori utama (main memori) dengan alat penyimpanan skunder atau alat input/output seperti printer, terminal, dan

card reader/punch. Pada kebanyakan sistem komputer, CPU tidak dibebani menangani tugas yang berhubungan dengan I/O. Tetapi, tanggung jawab untuk kontrol peralatan diserahkan pada prosesor I/O, yang dikenal sebagai saluran I/O. Saluran I/O itu sendiri merupakan prosesor yang sudah diprogram. Program-program yang di execute ini disebut channel program. Channel program ini menentukan operasi yang diperlukan untuk akses peralatan dan mengontrol jalur data (data pathway). Sistem operasi memberikan rutin standard yang digunakan untuk menjalankan saluran I/O.

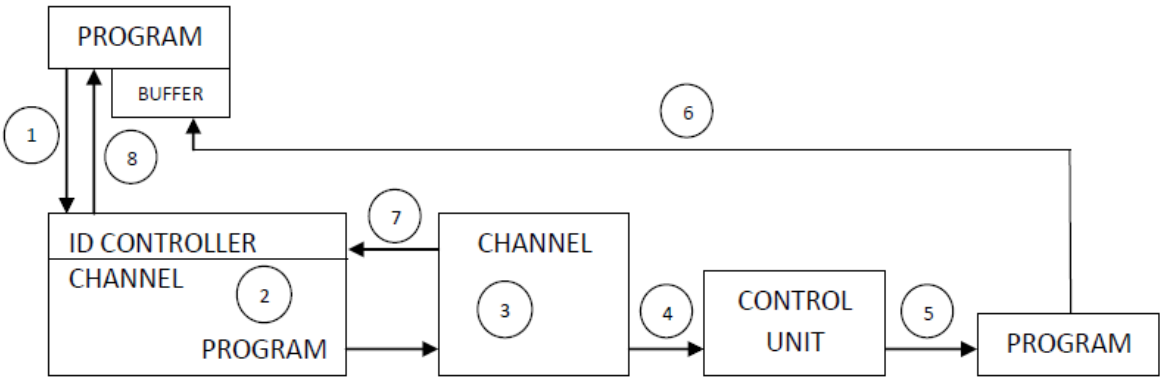


C. MANAJEMEN SALURAN

Tujuan dari saluran I/O yang bertindak sebagai perantara antara CPU-memori utama dengan unit pengontrol penyimpan. CPU berkomunikasi dengan saluran melalui beberapa perintah yang sederhana. Beberapa saluran akan member perintah seperti:

1. Test I/O, untuk menentukan apakah jalur (pathway) yang menuju peralatan sedang sibuk.
2. Start I/O, pada peralatan tertentu.
3. Halt I/O, pada peralatan tertentu.

Saluran biasanya berkomunikasi dengan CPU melalui cara interupsi. Interupsi akan terjadi jika keadaan error terdeteksi, misalnya interuksi CPU yang salah atau jika aktivitas I/O telah diakhiri. Jika interupsi terjadi, kontrol akan bercabang melalui rutin pengendali interupsi (interrupt-handler routine), dimana kontrol akan menentukan penyebab dari intertupsi, melakukan kegiatan yang tepat, kemudian mengembalikan kontrol pada pemanggil (caller). Jika sebuah program membutuhkan READ dari suatu berkas, maka serangkaian peristiwa pada gambar 3 ini akan terjadi.

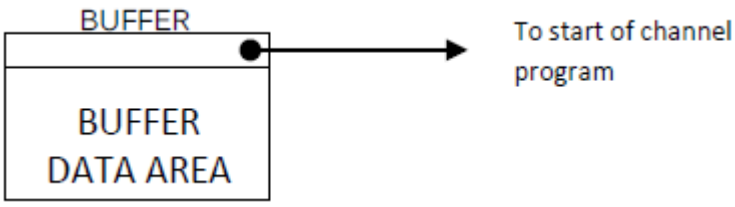


1. Program mengeluarkan sebuah READ, yang menginterupsi pengontrol I/O.
2. Pengontrol I/O membuat sebuah saluran program pada memori utama.
3. Saluran program dibaca dan dieksekusi oleh pemanggil saluran.
4. Sinyal yang tepat akan ditransmisikan ke pemanggil unit kontrol.
5. Sinyal ini diterjemahkan oleh unit kontrol dan digunakan untuk mengontrol peralatan operasi untuk membaca data yang diminta.
6. Data yang diminta akan mengalir dari peralatan sepanjang jalur (pathway) ke daerah penampung berkas (file buffer area) dalam ruang memori utama.
7. Interupsi yang dikeluarkan oleh saluran, digunakan untuk meneruskan sinyal pada waktu eksekusi program.
8. Kontrol kembali ke program.

D. MANAJEMEN BUFFER

1. Single Buffering

Gambar dibawah ini menunjukkan struktur data dari buffer dalam bentuk yang sederhana, yang terdiri dari satu record perblok dan satu buffer per berkas, dimana buffer ini berfungsi mengisikan permintaan dari sebuah program. Struktur buffer ini berisi sebuah pointer pada alamat awal dan channel program untuk berkas yang sedang dioperasikan.



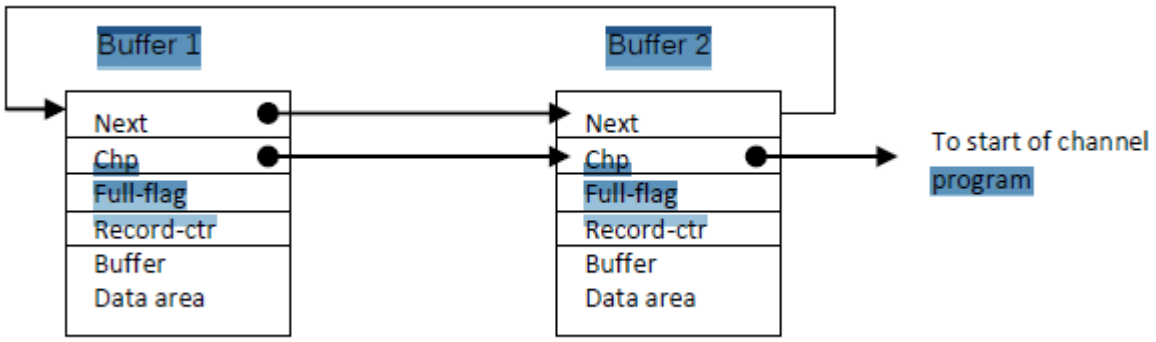
Struktur dasar dari channel program untuk mengisi buffer adalah:

- Tunggu instruksi READ dari program
- Memberitahukan instruksi start I/O ke kontrol unit
- Tunggu hingga buffer dikosongkan
- Memberitahukan interupsi pada program sehingga dapat mulai membaca dari buffer.

2. Double Buffering

Untuk mengurangi kemungkinan dari program menunggu, double buffering dapat digunakan. Dua dari tempat buffer yang ada, hanya satu yang ditetapkan untuk berkas. Ide dasar dari double buffering adalah jika consumer mengosongkan salah satu buffer, maka producer dapat mengisikan ke dalam buffer yang lain, pada saat buffer pertama sudah kosong, maka buffer yang kedua harus dalam keadaan penuh. Kemudian consumer dapat mengosongkan buffer kedua, pada saat producer mengisi buffer pertama, demikian seterusnya.

Struktur buffer untuk double buffering terdiri dari sebuah pointer yang menunjuk ke buffer berikutnya. Pada gambar di bawah ini. Adanya printer ini memungkinkan rutin producer dan consumer menjadi hal yang sangat umum.



Full-flag = { 0 jika buffer kosong atau sedang diisi
1 jika buffer penuh atau sedang dikosongkan;

Record-ctr = 1, ..., N