講 座

On-Line Real Time System의 槪念(Ⅲ)

朴 永 文*

——차 례——

〈前號 (Vol.23 No.2) p.27에서 계속〉

4. On-Line Real Time System의 Software 및 Memory 관리

이 節에서는 on-line real time system의 operating system 즉 software와 memory의 관리 (management)에 관하여 記述하기로 한다.

4—1 Operating System

on-line real time system의 운용은 操作員의 介入을 되도록이면 排除한 狀態에서 system program의 自動的인 處理에 依하여 達成된다. 이 system program은 executive system, monitor, supervisor (supervisory program) 等의 名稱으로 불려지는 control program과 이 밖에 language processor, service program, support program 等으로 구성되며, 適用業務에 따라 作成된 user program 또는 application program과 關聯되어 software system을 形成하게 되며, 이들 software system 一切를 operating system이라 부른다.

그런데 이 operating system은 크게 分類하여 control program과 處理用 program으로 區分되어, control program이 實行되는 狀態를 control mode 또는 supervisor mode라 하고, 處理用 program이 實行되는 狀態를 program mode 또는 user mode라 한다.

a) Control Program

control program은 system동작을 제어, 감시, 관리하는 program으로서, 入出力제어, 자원(resources)의 할당, interruption處理, scheduling, program과 데어터 領域의 관리, system의 維持 및 異常時 處理를 行하는 機能을 갖는다. 이들 機能을 要約하면 job control, task control, data control의 4가지 機能으로 分類된다.

b) 處理用 program

(ㄱ) Language Processor

language procesor는 user가 作成한 source program을 計算機가 直接 處理할 수 있는 object program으로 번역을 行하는 system program을 意味하며, FORTRAN, ALGOL, COBOL, PL/1 等의 compiler language로 作成된 source program을 번역하는 FORTRAN compiler, COBOL compiler, ALGOL compiler, PL/1 compiler와 assembly language로 作成된 source program을 번역하는 assembler 等 여러 種類가 있다.

(ㄴ) Service Program

日常 자주 使用되는 routine은 미리 計算機 製作會社에서 system program의 一部로서 開發하여 user의 부담을 덜어주고 있다. 즉 이런 routine을 service program 또는 utility program이라 부르며 分類/組

* 正會員 : 서울工大 副敎授(工學博士)

合 program, 各種 file取扱 program, link/edit program 等이 이에 屬한다.

(c) Supporc Program

Sysem의 正常運轉時에는 別로 使用되지 아니하나, 各種 테스트用 program, 진단 prgram, 端末 simulator 等 system의 設置, 圓滑한 維持에 必要한 program도 준비되어 있어야 하는데, 이와 같은 program을 support program이라한다.

(d) Application Program

各 transaction의 要求에 따라 control program의 制御下에 message를 實際로 處理하거나, 問題를 푸는데 使用되는 program을 말하며, 이 program은 system의 目的, 適用業務에 따라 다르기 때문에 user에 依하여 供給되며, 使用 language는 一般的으로 control program 및 language processor에 依하여 規制된다.

(e) User Program

通常的인 on-line system에서는 別로 使用되지 아니하나, time-sharing system의 加込者 또는 研究所 教育機關에서 端末裝置로부터 user가 作成한 program의 傳送, 實行 및 回答을 要求하는 경우가 있다. 이 경우의 program을 user program이라 한다.

4—2. Memory 관리

on-line real time system에서의 memory관리문제와 中央處理裝置와 端末 또는 其他 入出力裝置間의 情報 送受時의 core memory內의 入出力領域 確保問題는 보통의 一括處理(batch processing)의 경우와는 比較가 되지 않을 程度로 복잡하고 重要하다.

a) Memory의 動的割當(Dynamic Allocation)

real time system에서는 message가 不規則 적으로 들어 올 뿐만 아니라, multi-programming技法을 使用하는 관계로, 여러 個의 active program (core memory內에 存在하는 user program 또는 application program)이 core memory內에 共存하므로, 이에 必要한 記憶場所를 미리 固定시킨다는 것은 memory의 利用效率의 觀點에서 바람직하지 못하다. 따라서, 일반적으로 memory를 一定크기의 block로 分割하여, 이 block單位로 割當 使用하고, 使用後에는 이를 解放하는 技法을 採用하고 있다. 즉, 아직 使用하지 않은 (즉 使用可能한) 블록을 사슬(chain) 결합하는데, 결합된 各 block에는 다음 결합番地語(pointer word)를 設定하여 다음 block의 番地를 알게 하고, 先頭 block의 番地를 指示하기 위하여 別途로 基準番地語(base word)를 設定한다. 이와 같은 使用可能 block의 결합狀態表를 使用可能블록表(available block list)라 부른다.

memory割當의 要求가 있을 경우에는 이 使用可能 블록表를 參照함으로써 基準番地語가 指示하는 先頭의 block부터 順次的으로 必要한 個數의 block만큼 memory를 割當하고 나머지 先頭 block의 番地를 基準番地로 代替한다. 또 使用이 完了된 block은 使用可能블록表의 先頭에 附加한다. block의 割當이나 解放은 macro命令을 通하여 control program이 行하게 되는데, 이와 같은 技法을 memory의 動的割當技法이라고 한다.

on-linc real time system에서는 core memory에 常駐하지 아니하는 application program, user program이나 control program의 부분이 많고, pragram의 길이도 크므로, 이들을 block의 크기만큼의 小部分 즉 segment로 分割한 形態로 준비되고 있다. core memory에 들어 있지 아니하는 program이나 program segment에 對한 要求가 있으면, 使用可能블록表를 參照함으로써 前述한 바와 같은 方法으로 必要한 數만큼의 block이 割當된 後 所要의 segment가 file(補助記憶裝置)로부터 이 block에 읽어 들여진다. 이때 使用되는 block의 番地는 豫測이 不可能하므로 program은 relocatable한 形式으로 (平行移動可能한 相對番地形式으로) 作成되어 있어야 한다.

multi-programming技法을 採用하는 on-line real time system에서는 memory의 保護問題도 또한 重要하다. program의 잘못 作成, 損傷, hardware上의 故障等으로 어느 한 program이 다른 領域內의 block에 침입하여 情報를 파괴할 우려가 있다. 이와 같은 可能性을 피하기 위한 方法으로는 上, 下限 register에 依한 方法, 保護 key에 依한 方法 等이 있으나, 여기서는 詳論하지 않기로 한다.

b) Page技法

time-sharing system에서는 memory의 動的割當에 있어서 더 進一步한 page技法을 採用하고 있는데, 이 技法은 최근 time-sharing system이 아닌 보통의 大形 電子計算機에도 점차로 導入되고 있다.

이 方式에서는 그림 18에 표시한 바와 같이, block變換機構인 hardware가 block單位로 user가 使用하는 論理的 block番號를 實際의 物理的 番號로 變換하게 되어 있다.

user가 使用하는 語는 page番號와 그 page內의 場所를 나타내는 番號로 指定되며, page番號는 上述한 變換機構에 依하여 實際로 core memory內의 block番號로 變換되고, 이 block番號와 block內의 場所(즉 page內의 番號)를 指示하는 line番號와 並合되어 實際로 core memory內의 物理的 場所를 指定하게 된다. 각 user에 對하여 page番號와 block番號가 1對1로 對應하는 變換表가 마련되어 있으며, 이 管理는 control

program이 行한다. 이와같이 變換狀態의 記憶은 processor의 hardware上에 register形式으로 이루어지거나, 또는 core memory內의 어느 領域에 이루어지게 된다.

이 方式에 依하면, user가 利用하는 記憶空間은 實際로 存在하는 core memory의 物理的 空間에 依하여 制約을 받음이 없이, 거의 無限으로 擴大될 수 있으므로 user의 立場에서는 外部記憶裝置의 記憶空間도 core memory의 記憶空間과 同一하게 取扱될 수 있다.) 實際의 物理的 空間을 거의 意識할 必要가 없다.

그런데 이 方式에 依하면, 變換表가 너무 커지고, program作成과 debugging의 觀點에서도 program을 보다 적은 單位로 分割할 必要가 있으므로, page 番號를 다시 2部分으로 分割하여 階層化하는 것이 便

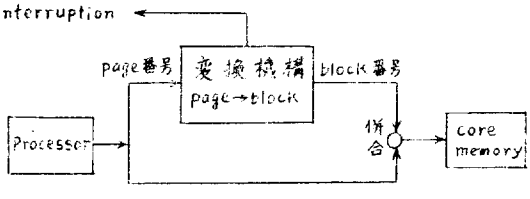


그림 18. Page technique.

利할 경우가 있다. 이 경우에는 2部分으로 分割된 上位의 값은 下位의 값이 들어 있는 block을 指定하고, 이 中에서 實際로 program이 들어 있는 block을 指示할수가 있다. 이 方式을 採用할 경우, 하나의 program은 다수의 segment로 構成되고, 이 segment는 하나의 獨立된 program 單位로 取扱되어, 다시 page로 細分된다. segment의 指定은 特定의 register에 segment名稱을 set함으로써 이루어지고, 각 segment에 對하여는 上述한 바와 같은 page變換表가 作成되는 式의 階層的 變換이 이루어진다. 이와 같은 方式을 2次元番地指定方式이라고 부른다.

5. On-Line Real Time System의 Bufferring Interruption 및 Seheduling

on-line real time system에서는 multi-programming技法으로 transaction이 處理되는 것이 普通인데, 이 原則을 把握하기 위하여는 入出力 bufferring, interruption 및 scheduling에 관한 基本的 槪念을 理解해 둘 必要가 있다.

5—1. 入出力 Bufferring

中央處理裝置와 端末 또는 其他 入出力裝置 間의 情報의 送受信에는 入出力 領域으로서 core memory가 使用되는데 이 領域을 入出力 bufferring이라고 부른다. 一般的으로 buffer는 情報의 傳送速度가 다른 兩傳送系 또는 直列 傳送系와 並列 傳送系를 接續하여 하나의 傳送系를 形成할 경우, 傳送速度를 同期化시킴으로써 傳送能率이 좋은 傳送系側의 時間的 負擔을 輕減시키는 役割을 遂行한다. 補助記憶裝置나 또는 printer. card 等의 入出力裝置와 中央處理裝置 間의 入出力 buffer는 入出力速度가 타이프라이터式 端末裝置의 경우보다 빠르고 1回分 情報의 傳送量도 比較的 적으므로, buffer의 容量이나 使用條件面에서 그다지 問題가 없겠으나, 端末裝置와 中央處理裝置間에 message變換을 行하는 on-line system의 入出力 buffer는 고객의 要求에 反應하여 가면서 端末에서 message를 送信할 뿐만 아니라, 多數의 고객의 要求가 一時에 集中하는 不規則(random) 現象이 發生하는 경우가 많으므로, 前者의 경우보다 더욱 철저한 hardware上의 配慮가 必要하게 된다. 보통 使用되는 入出力 buffer로서는 다음 두 種類를 들 수 있다.

a) 專用 Buffer

專用 buffer는 各 回線에 1對1로 對應하는 一定길이의 core memory로서, 그 길이는 그 回線에서 送受信되는 最大 길이의 message量에 따라 決定된다. 이 方法은 message의 길이가 짧거나, 固定길이의 message만을 取扱하는 경우에 適合하다.

b) 動的 buffer

message의 길이가 길거나, 時時刻刻으로 길이가 變動하는 경우에는 buffer의 使用效率을 높이기 위하여 全 回路가 共同으로 使用하는 buffer를 設置한다. 이와같은 buffer를 動的 buffer라고 부른다.

動的 buffer에서는 memory의 動的割當의 경우와 마찬가지로 core領域을 一定 길이의 block으로 細分하여, 이 block들로서 構成된 buffer pool이 形成된다.

送受信될 message를 위한 buffer割當의 要求가 있으면, buffer pool에서 必要한 數만큼의 block이 割當된다. 大部分의 경우, 動的 buffer에서는 message의 送受信時 一時的으로 蓄積된 情報는 곧 drum이나 disk에 옮겨진다.

5—2. Interruption

어떠한 原因 或은 狀態가 發生하면, 現在까지 實行(execute)하고 있는 program의 處理가 一時 中斷되고 다음 順序는 control program에 맡겨진다. 그러면 control program은 中斷된 program의 再開에 必要한 情報를 適當한 곳에 待避시킨 後, 中斷의 原因에 對應하는 處理루우틴(interrupt service routine)을 動作시켜 適切한 措置를 講求하거나, 기다리고 있는 다른 active program을 實行시킨다. 그러다가 中斷의

原因이 消滅되면 이미 待避시켜 둔 情報를 되돌려 놓고 中斷된 program에 control을 다시 옮긴다. 그러면 中斷된 program은 處理를 續行한다. 이와 같은 過程을 interruption이라고 부른다.

interruption은 보통 다음과 같은 狀態가 發生하였을 경우 이루어진다.

① 入出力動作의 開始
② 入出力動作이나 file (補助記憶裝置)動作의 完了
③ error나 異常狀態의 檢出
④ 時刻起動處理의 開始
⑤ 操作員의 介入

interruption의 狀態가 여러個 存在할 경우에는 system의 維持上 處理의 優先順位에 따라 interruption이 處理되는데, on-line real time system에서는 모두이 目的을 위한 多段階(multi-level) interruption機構가 마련되어 있다. 예를 든다면, 入力 transaction을 위한 interruption處理中, 中央處理裝置에 異常狀態가 發生하면 後者의 處理가 優先된다.

5—3. Scheduling

multi-programming技法을 採用하는 on-line real time system에서는 system內에 여러個의 transaction이 共存하여 處理를 기다리고 있으므로, 入出力側, 記憶媒體內, channel側 等에서 待期行列이 이루어진다. 즉 transaction의 到着順序에 따라 入力 待期行列(input quene), channel待期 行列(channel waiting quene), 中央處理裝置 待期 行列(CPU quene), 出力待期行列(output quene)이 이루어 진다. 그런데, 待期 行列의 處理 順序는 channel待期의 경우를 例外로 한다면, 先着順(first-in first-out) 또는 優先順(priority)을 고려한 先着順으로 決定된다.

어느 한 處理가 끝났을 경우, 다음 處理順序로서 여러個의 行列中 어느 行列의 transaction을 處理한 것인가를 決定하는 것은 control program이 行하여야 할 課題이다. control program中 이와 같은 處理順序를 決定하는 루우틴을 main scheduler 또는 CPU loop라고 부르며, scheduling이 algorithm은 system의 크기, 處理 program의 優先順位, 應答時間, 使用機器에 따라 各其 다르다. 단, channel 待期行列이나 端末裝置에 對한 message의 送出, 送入에 對한 處理順序는 外部機器의 使用條件에 따라 決定되는 別途의 I/O scheduler에 의하여 決定된다. 보통 I/O scheduler는 入出力機器나 file(補助記憶裝置)의 動作이 끝나서 interruption이 發生하면, 곧 다음 行하여야 할 入出力動作을 선택하여 이를 開始하도록 한다.

6. On-Line Real Time System의 File

on-line real time system의 data file이 갖추어야 할 基本的 要件으로서는 ① 大容量性, ② 高速性, ③ 信賴性 및 ④ 經濟性을 들 수 있는 바, process control system의 경우를 除外한 大部分의 system에서는 多量의 data를 集中管理하는 관계로 大容量의 random access file裝置를 必要로 하며, 個個의 處理에 對한 file을 갖기 보다는 많은 處理에 對하여 共通的으로 使用하는 하나의 data base를 갖는 것이 바람직하며, 實際로 業務의 進行과 더불어 不規則하게 發生하는 transaction에 對한 應答特性이 優先하여야 하며, 또 이와 못지 않게 system의 信賴性과 經濟性의 圖謀가 極히 重要한 問題로 登場하게 된다. 따라서 이와 같은 要件을 염두에 두면서, 이 節에서는 on-line real time system의 file裝置의 種類, 構成, record의 番地指定方式(addressing system)等에 對하여 略述하기로 한다.

6—1. File裝置의 種類

一般的으로 on-line real time system에서 使用되는 file裝置의 特性을 評價하는 가장 重要한 因子는 平均 access 時間, 傳送速度, 記憶容量 및 그 經濟性인 바, 裝置選擇時에는 이 點을 아울러 考慮하여야 한다. 一般的으로 말해서, file裝置는 階層構造를 이루고 있어, 重要한 順序로 羅列하면, core memory, drum (또는 大容量 低價格 core memory), disk, data cell順位가 된다.

例를 들면, 座席豫約의 경우, 出發前 1週間分의 豫約 record는 drum (또는 disk)에 收容하고, 나머지 record는 disk (또는 data cell)에 收容하고, transaction의 處理作業은 core memory에서 行하는 것 等이다.

6—2. File의 構成

transaction의 發生時, file中의 수많은 record中에서 必要한 record만을 高速으로 뽑아 내기 위하여는 이에 알맞는 file構造와 番地指定方式을 採擇하여야 한다.

그런데 real time system에서 하나의 data base는 몇 個의 file로서 이루어져 있으며, 또 이 file은 다시 다음과 같은 階層構造를 이루고 있다. 즉, file의 構造로서는 나무構造(tree structure)와, 格子構造(lattice structure)가 있는데, 나무構造는 그림 19에 表示한 바와 같이, 나무가지처럼 하나의 根結點(root node)에서 여러個의 部分 나무가지가 분기되어 나가는 形態를 말하며, 여기서 各 結點(node)은 즉 file을 나타내고,

한 結點 즉 file에서 분기되는 보다 下位의 結點은 部分 file이라 불려지고, 最下位의 結點에 對應하는 file은 많은 record로서 구성된다. 例를 들면, 最上位의 file을 乘客 file로 보면, 그 다음 file은 日字 file, 다시 乘便 file, 乘客 record順으로 내려간다.

경우에 따라서는 그림 20에 表示한 바와 같이, file A를 file F와 file G에서 兩 方向에서 接近하는 것이 바람직한 경우도 있다. 이와 같이, 어느 特定의 record를 多重으로 몇 個의 file에 登錄하는 file의 構造를 file의 格子構造라고 부른다.

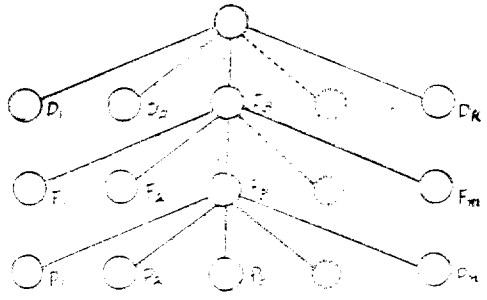


그림 19. Tree structure of files.

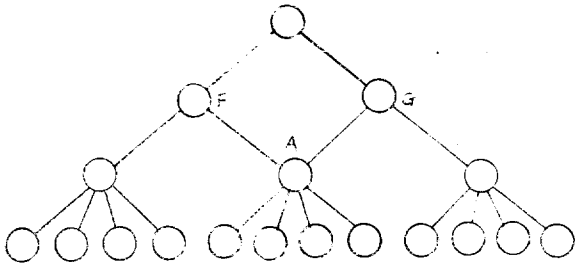


그림 20. Lattice structure of files.

個個의 file을 識別하기 위하여는 名稱이나 key가 附與되는데, 이 名稱이나 key는 file의 論理的 構造에 對應한다. 그런데, 이 file이나 record가 實際로 記憶되는 物理的 場所(physical location) 즉 番地(address)와 名稱 또는 key를 對應시킬 必要가 있다. 이를 위한 가장 一般的인 方法은 索引表(directory)에 依한다. 그림 21은 이 索引表의 概念을 圖示한 것으로, 지금 3段階의 水準(level)으로 이루어진 file의 나무 構造를 假定하고, file A의 部分 file B에 屬하는 record C의 番地를 찾으려 할 때, 이에 對應하는 3段階 索引表를 階層的으로 作成한다. 根表에는 根 file의 名稱과 이에 對應하는 第2 段階 指示子(pointer)가 들어 있고, 또 第2段階의 索引表에는 第2段階 file의 名稱과 이에 對應하는 第3段階의 指示子가 들어 있고, 마지막 段階의 索引表에는 record의 名稱과 이에 對應하는 番地가 들어 있다. 따라서 A, B, C의 番地를 얻기 위하여는 根表에서 A를 찾아서 指示子 α를 얻고, α에 依하여 A表를 찾고, A表에서 B를 찾아서 β를 얻고, β에 依하여 B表에 이르러 C를 찾아서 번지 γ를 얻게 된다.

各 索引表는 對應 file의 첫머리에 두는 경우도 있고, 獨立的으로 全 索引表를 한데 모아서 다른 補助記憶裝置에 收容하는 경우도 있다.

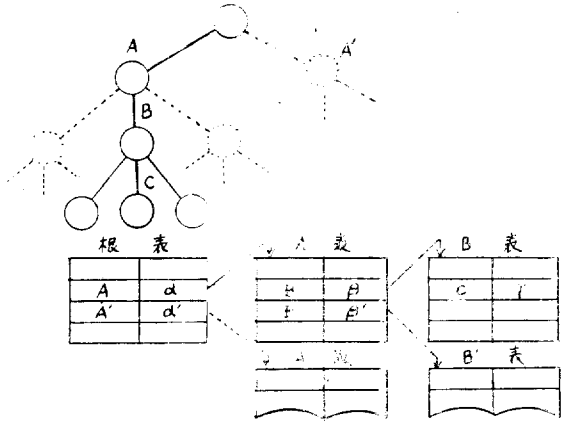
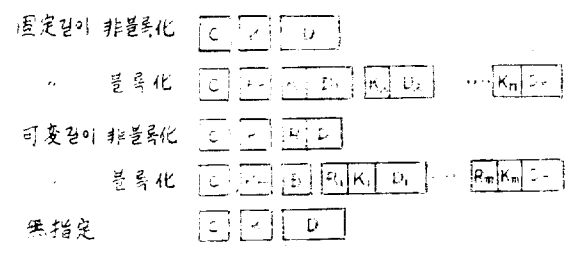


그림 21. Record access directory.

6—3. Record의 形式

file作成時 여러 個의 record를 묶어서 1個의 block을 形成하는 경우가 많은데, 이 操作을 블록化(blocking)라하고, 반대로 읽어 낼 때 block에서 record를 分離하는 操作을 逆블록化한다. 그리고, 처음부터 블록化하지 아니하는 것을 非블록化라 한다.

그림 22는 record의 各種 形式을 圖示한 것인데, record에는 data 以外에 이를 識別하기 위한 名稱 또는 key와 制御情報가 附加된다. 制御情報領域에는 計數領域(count area), key領域, block길이領域, record 길이領域의 一部또는 全部가 包含된다. 그리고, 各 record의 앞에는 record의 開始를 가리키는 마아크가, 또 各 領域 사이에는 間隙(gap)이 hardware上에 設定된다. 普通은 製造會社에서 마련한 control program 中에 이와같은 data管理機能을 맡는 루우틴(routine)이 있으나, 特히 無指定 record의 作成時에는 使用者가 이를 위하여 獨自的인 管理루우틴을 함께 作成하여야 한다.



C : 計數領域 R : record길이領域 k : key領域
D : data領域 B : 블록길이領域

그림 22. Record formats.

6—4. File의 編成 方式

file의 編成 方式은 裝置, 內容, 處理目的에 따라 各其 다르겠으나, 여기서는 그 代表的 方式 몇 種類에 對하여 言及하기로 한다.

a) 直列(sequential)編成

이 方式은 record가 key의 順序나 그 밖의 어떤 順序에 따라 記憶媒體에 記錄되는 方式으로, 磁氣테이프, 카아드等의 경우에 자주 使用되며, record의 選擇을 위하여는 file의 探索이 必要하다. 따라서 이 方式은 探索에 所要되는 時間이 너무 길어 on-line real time system의 transaction處理에는 不適合하다.

b) 分割直列(partial sequential)編成

file을 構成하는 record가 몇 個의 組로 分割되고, 같은 組內에서는 前述한 바와 같은 直列配列되는 方式으로, 이 方式은 보통 program, subroutine, 表 等의 file에 자주 使用된다.

c) 索引(indexed sequential)編成

이 方式은 key의 順序에 따라 record가 直列的으로 配列되어 있을 뿐만 아니라, 이 record가 들어 있는 track 또는 이 track이 들어 있는 cylinder 等 記憶媒體의 構成 單位에 對하여도 key의 順序에 따라 track 또는 cylinder의 key와 그 番地를 對應시킨 索引表가 마련되어 있어 이 索引表를 參照함으로써 個個의 record에 access할 수 있는 方式으로, 이 方式에 依하면, record를 直列處理를 할 수도 있고, 또 個個의 record에 比較的 迅速하게 access할 수 있는 利點이 있다.

d) 無順(random)編成

이 方式은 record의 配列에는 順序가 없는 代身, key와 記憶媒體上의 record 番地間에는 어떠한 對應關係를 갖도록 한 編成方式을 말하며, 이 對應 關係는 key變換法 또는 索引表 等에 의하여 맺어진다. 이 方式은 record의 無順 處理에 適合하며, 特히 個個의 record에의 access時間에 嚴格한 制約이 주어지는 real time處理를 위하여는 가장 有力한 方式이다.

이 경우, file을 編成하거나, 編成된 file의 record를 읽어 내거나, 써 넣을 경우, 使用者가 data管理 macro命令을 使用하게 되는데, 이 macro命令을 access macro라고 부른다.

6—5. File處理의 經濟性 및 信賴性

file裝置가 주어졌을 경우, file의 使用空間을 最小限으로 줄여서 file處理의 經濟性을 높이는 問題는 매우 重要하다.

그 具體的 方法으로서는, 이름 等의 識別子(identifier)를 코오드(code)로 變換하거나 10進 數値를 2進數値로 變換하는 record壓縮法, 몇 個의 record를 1個의 物理的 record로 묶는 block化 法 等을 들수 있다. 그러나, real time system에서는 이 目的을 達成하기 위한 餘分의 program 및 file이 必要할 뿐만 아니라, channel占有 時間이 길어져, 오히려 全體的인 效率이 低下할 수도 있으므로, 이들 方法의 採用時에는 그 得失에 對한 愼重한 檢討가 要請된다.

다음으로는, on-line real system file의 信賴性 改善에 關한 問題이다. file裝置를 2組 準備하여 兩者의 內容을 같게 함으로써 file의 正確性을 체크하거나 또는 어느 한 組가 破損될 경우 다른 組를 使用함으로써 信賴度를 向上할수 있다. 그러나 經濟的 觀點에서 2重投資가 어려울 경우에는, 한 쪽의 裝置에는 集約된 內容만을 記憶해 두는 方法도 생각할 수 있다. 더욱 經濟的인 方法으로는 random access file 즉 disk나 drum은 1組만 두되, 更新된(updated) transaction의 記錄을 磁氣테이프에 옮겨 놓고, 하루의 處理 終了後 磁氣테이프에 記錄된 transaction으로 부터 當日의 file 更新을 再現하여 그 結果를 比較해 보는 方法도 생각할 수 있다.

더욱 一般的인 方法으로서는, 後備裝置(backup memory)를 準備해 두는 方式이다. 例를 들면, real time處理에서 core memory에 使用되는 表가 어떠한 原因으로 파괴되는 경우에 對備하여, 한 段階 낮은 水準의 記憶裝置인 高速 drum에 본떠 놓고, 또 drum에서 읽어 내는 航空便의 時間表 等은 이보다 낮은 水準인 disk에 본떠 놓는다. 經濟的인 理由로서 이와 같은 2重化도 어려울 경우에는 定期的으로 어느 時點의 狀態를 dump한 것과 그 時點 以後 file을 更新한 transaction의 記錄을 比較함으로써 後備復旧을 行하는 方法도 생각할 수 있다.

7. On-Line Real Time System의 信賴性

real time system의 信賴性은 system에 依한 서비스의 連續性과 서비스의 正確性의 兩側面에 評價되어야 한다. 一般的으로 on-line real time system은 24時間 稼動되는 경우가 많으므로, 높은 稼動率을 維持하기 위한 여러 가지 技術的 方法이 採用되어야 하며, 서비스의 正確性에 對하여는 특히 file을 中心으로 한 system의 正確性에 重點을 둘 必要가 있다.

信賴度의 一般的 定義는 **"주어진 動作條件下에 豫期된 期間中 正確하게 주어진 目的을 達成하는 確率(probability)"**이다.

따라서, 이를 計量的으로 評價하기 위한 尺度도 아울러 定義되어야 한다. 現在까지 採用되고 있는 信賴度 評價尺度로서는 ① system稼動率(availability), ② 平均故障時間(MTBF, mean time between failure),

〈p.34에 계속〉