技術解說 1

Time Sharing System

김 기 룡*

1. 序 論

高性能의 大型計算機를 不特定多數의 利用者에 依하여 언제 어디서나 즉시 利用할 수 있다는 것은 使用者에게는 理想的 시스템이라 할 수 있다. 그러나 이와같은 콤퓨터의 利用方法은 오래전 부터 電子工學者들 사이에 槪念的으로 應用되는 時分割(Time Sharing)의 사상을 計算機의 利用方式에 採用함으로서 實現된 것이다. 計算機의 Time Sharing에 대하여는 1961년에 美國에서 John McCarthy 教授에 依하여 發表되어 F. J. Corbato의 指導로 開發된 M. I. T. 의 計算센타에서 파이로트·모델(시험모델)의 最初의 시험이 이루어졌다. 이것은 CTSS(Compatible Time Sharing System)이라 불리는 시스템으로 大型타임쉐어링시스템의 最初의 것으로 M. I. T. 의 MAC (Machine Aided Cognition) 計劃下에 1963년 부터 MAC 시스템으로 利用이 始作되었다. 이 시스템에는 基本的 터미날로 52의 텔레타이프와 56개의 셀렉트릭·텔레타이프라이터가 있고 전화회선을 經由하여 CPU IBM 7094와 連結되어 있다. 또 特別한 멀티플 CRT 디스플레이와 라이트 펜, 文字發生器가 CPU와 直接連結되어 CAD (Computer aided design) 計劃에 使用되고 있다. 또한 CPU에 1200보의 回線으로 接續되어 있는 計算機 PDP-6에 依해서 遠隔地點에서의 圖形表示시스템의 使用을 可能케 하고 있다.

메모리는 64K 語의 코아메모리를 中心으로 185 K 語의 드럼·메모리가 2臺, 18,000K 語의 記憶領域을 갖는 디스크·파일 2臺로 코아메모리를 백·업하고 있다. 64K 語의 코아는 32K 語씩 둘로 나누어 그 하나는 user의 領域으로 使用하고 다른 하나는 타임·쉐어링·시스템의 슈퍼봐이서·프로그램으로 使用하고 있다. 드럼은 다른 프로그램을 實行할 場所를 마련하도록 코아메모리에서 드럼으로 프로그램의 退避, 交換을 하는 目的으로 使用된다. 또 디스크·파일은 user의 앞에 있는 콘솔에서 直接 大量의 넓은 意味의 데이타를 파일하고 액세스(接)할수 있다. 파일의 內容으로는 콤파일러群, 많은 共用프로그램 使用者의 個人파일情報, 그리고 user의 프로그램등이 있고 恒常 프로그래밍에 對한 새롭고 便利한 소프트웨어, 새 言語, 필요한 서브루으틴등을 user의 성장과 더부러 파일에 追加해가는 것으로 시스템의 顯著한 特徵을 이루고 있다.

2. 타임쉐어링

計算機의 演算速度가 늦어 平均命令實行時間이 밀리秒·程度의 初期에는 user가 콘솔앞에 앉아서 CPU의 動作狀態를 나타내는 램프의 點滅을 드려다보고 있었다. 어떤 루우틴에서 다른 루우틴으로 비약하고, 그 곳에서 연산하고 다시 루우틴으로 되돌아온다는 일들이 램프의 모양으로 알 수 있고, 演算이 빨리 끝나기를 바라고 結果가 기다려졌다. 그러나 演算速度가 조금 上昇하여 두자리 마이크로秒程度가 되면, 入出力(I/O) 裝置의 動力速度가 늦게 느껴지고, CPU로 演算을 續行하려해도 I/O 動作이 完了되지 않으므로 다음 演算過程으로 가지못한다는 理由로 CPU의 쉬는 時間이 顯著해졌다. 그리하여 CPU의 遊休時間을 有效하게 利用하려는 생각으로, I/O 裝置의 busy 狀態를 檢出하여 busy 時에는 다른 하나의 獨立된 프로그램의 實行에 CPU를 使用한다는 멀티플프로그래밍의 手法이 取해지게 되었다. 그

* 崇田大學 電子計算所長

러나 멀티플프로그래밍으로의 處理狀況을 調査하면 並行處理되는 일들 사이에 어느 일에서 다른 일에의 移行은, 일實行中에 하는 入出力動作에 依해서 busy 狀態가 發生하여, 分割이 발생하므로 조합되는 Job의 性質에 따라서 어느 일이 먼저 完了되든가 또는 모든 일에 CPU 타임을 고르게 割當할수는 없다. 따라서 不特定多數의 使用者의 여러가지 일에 對하여 同時에 均等한 計算處理를 하기 위하여는 CPU 타임을 細分하여 單位時間을 各各의 일에 適當히 分配하여야 되리라고 생각된다. CPU의 速度가 1마이크로秒로된 現在 타이쉐어링·시스템에서는 미리程度의 精度로 任意의 時間間隔으로 프로그램에 끼어들도록 하드웨어인 인터발·타이머로 CPU 타임의 單位時間(quantum time)을 複數의 일로 分配使用케 한다. 即 强制的으로 끼어들기 위하여 實行中의 프로그램을 中斷하고, 어느 일에서 다른 일로 CPU의 使用配當의 變更을 시키는 點이 멀티프로그래밍과 本質的으로 다르다. 또 複數의 일이 섞여있고 並行하여 處理가 進行되는 것은 CPU의 時分割使用外에 코아메모리의 共用, 나아가서 파일의 共用등의 問題가 발생하게 된다. 다음으로 TSS의 主要한 特徵과 그 시스템設計에 있어서의 몇가지 問題에 對하여 논한다.

3. TSS의 規模

① 單能特殊用途의 TSS

限定된 言語시스템으로 프로그램을 作成하고 實行한다. 그리고 파일은 매우 작고 簡單한 파일構造로 주로 프로그램의 debug나 簡單한 計算에 使用된다. 例컨대 Rand Corporation의 TOSS 시스템이나 IBM의 Quiktran 시스템等이 이에 속한다.

② 汎用大型 TSS

이것은 다음과 같은 特徵을 갖고 있다.

㉠ 端末이 적어도 20개를 넘으며 많은 user에게 보다 빠른 應答時間으로 처리한다.

㉡ 파일내에 프로그램 수정용의 에디터, debugging의 機能, 아셈블러의 사용, 여러가지 豐富한 種類의 콘파이러의 準備, 應用프로그램의 라이브라리의 整備等이 行해진다.

㉢ user가 전용할 수있는 大容量記憶裝置에 依한 파일·시스템이 있고, 이 파일에는 쉽게 접속될 수 있고 또한 必要에 따라 各種情報가 공용된다.

㉣ TSS의 시스템자신이, 時間의 經過와 함께 向上 發展을 繼續해나가야 하므로 追加機能에 對하여 응할 수 있는 오픈엔디드의 設計로 되어 있다는 것 등이다.

그리고 이러한 規模의 시스템에는 이미 列擧한 CTSS, 또한 最近의 MAC 計劃에서의 MIT의 MULTICS(Multiplered Information and Computing Service) 시스템, SDC의 Q32 시스템, 캘리포니아大學 버클리의 XDS-940 시스템 등이 있다.

4. 어드레스 空間

限定된 코아메모리의 容量內에서 多數의 프로그램을 進行시키기 위하여는 이들 프로그램의 實行時에 놓여질 領域, 또한 이들이 必要로 하는 데이타記憶을 위한 領域의 管理가 重要해진다. 이때문에 어드레스에 對한 槪念으로 두개의 어드레스空間(address space)이 定義된다. 即 그 하나는 하드웨어가 갖는 番地, 絕對番地이고 휘지칼·어드레스(physical address)空間이라 불리우며, 다른 하나는 소프트웨어에서 주로 使用되는 이름으로, 로지칼한 어드레스인 네임·스페이스라고도 불리우는 로지칼·어드레스(logical address) 空間이다.

그리고 로지칼·어드레스空間이, 對象으로 하고 있는 휘지칼·어드레스空間보다 커졌을 境遇에는 버어튜얼(Virtual)메모리라고 부른다.

TSS의 프로세서는 主記憶裝置인 코어메모리, 드럼, 디스크등의 大容量의 補助記憶裝置를 包含하는 모든 휘지칼·어드레스空間에 對하여 로지칼·어드레스를 對應시킨다. 로지칼·어드레스空間에서 휘지칼·어드레스로의 變換은 하드웨어와 소프트·웨어의 兩面에서 行하여지게 된다.

5. 메모리프로텍션

이들 모든 記憶媒體上의 어드레스는 實際로는 多數의 使用者의 프로그램과 그 데이타領域, 나

아가서는 시스템自身의 프로그램이 들어가는 領域, 데이타領域등으로 分割되어 使用된다. 여기서 問題가 되는 것은, 이들 相互間의 記憶情報의 共同이용과 외부로부터의 攪亂, 引用等으로부터의 保護에 對한 시스템上의 處置이다. TSS를 實施하는 計算機에는 하드웨어로 記憶保護機構를 가추고 있고, 프로그램의 實行에 있어서, 그 아페란드·어드레스에 의해서 다음과 같은 保護가 이루어진다. 그 種類로는

1) 전적으로 액세스를 禁止한다.
2) 읽기 쓰기가 許容되는 경우
3) 읽기만 許容되는 경우
4) 實行만 許容되는 경우

등이 있다.

프로텍션은 휘지칼및 로지칼어드레스 스페이스의 양쪽에 대하여 각각 이루어진다. 그리고 후에 설명하는 다이나믹·리로케이션의 프로세스中에 恒時 프로텍션에 대한 체크가 이루어진다.

6. TSS로 同時에 處理되는 JOB·프로그램이 가져야할 性質

利用者의 일의 하나 하나는 TSS를 構成하는 시스템上의 制限을 넘지않는 範圍로 또 프로그램이 단순화되고 나아가서는 이들의 프로그램이 同時에 處理되고 시스템이 複數의 CPU를 가질때는 複數 CPU로 處理하기 쉽다는점이 要求된다.

또한 시스템·프로그램중의 팩케지(콤파일러, 에디터, 시스템·서브루으틴)등은 多數의 user로부터 同時에 같은 것이 利用되는 일이 자주 생긴다. 그러나 多數의 user가 각각 要求하는 프로그램을 自己의 領域內에 복사를 떠서 使用하는 것은 메모리의 각부분이 동일한 프로그램으로 점유 되어 不經濟이다. 그때문에 多數의 user가 單一個의 프로그램의 복사로 共同으로 할수있는 構造를 가진 프로그램形式으로 準備하여 이것을 리엔트란트(reentant)프로그램이라 부른다.

리엔트란트·프로그램은 實行中 自身의 프로그램을 다시 쓰는 일은 없다. 그 프로그램으로 使用하는 데이타領域은 프로그램自身에서 分離되어, 각 user의 작업이 實行時에 갖는 어드레스空間內에 一時記憶領域이 準備된다.

7. 스와핑(Swapping)

메인코아메모리를 多重으로 使用하기 위해 생기는 문제는, 코아메모리上의 情報와 로지칼·어드레스·스페이스의 大部分을 차지하는 補助記憶裝置로 擴張되어, 저장된 情報사이에 行해지는 交換(Swapping)이란 過程이다. 單純히 일이 코아위에 하나밖에 걸리지 않는 경우에, n個의 端末의 일에 對하여 各各 quantum·타임 q를 주고, n개의 일이 同時에 進行될 때, 매우 理想化된 關係에 있어서는 한개의 터어미날의 應答時間과 最大端末數 n_{max}와의 關係는 다음 式으로 표시된다.

$$n_{max}=\frac{t_r(1-\eta)}{2t_s+q} \qquad (1)$$

t_r은 應答時間, η는 시스템오버헤드로 차지한 比率이고, t_s는 코아에서 補助記憶으로, 또는 補助記憶에서 코아로의 一方向에의 傳送에 要하는 平均時間이다. 따라서 코아上에 複數의 Job를 상주시킬 때는, t_s의 값은 적게 할수 있다. 따라서 複數의 프로그램을 메인·메모리上에 바꾸어 놓으면서 차래로 配置하는 方法이 여러가지 考察되고 있다.

그중 가장 簡單한 것은 스태틱·리로케이션이다. 이것은 스와프 이전에 주어진 프로그램의 코아上의 저장開始番地와 그 프로그램이 補助記憶裝置로 옮겨저서 다시 코아上으로 되돌아왔을 때에 주어지는 格納開始番地가 같은 경우이다. 이 格納開始番地는 各各 휘지칼·어드레스이다. 그러나 이 方法은 코아上에 同時에 配置되는 각 일의 크기의 차이에서 발생하는 各 實行時에 있어서 메모리의 未使用領域의 管理에 柔軟性을 결여케 한다. 따라서 다음의 다이나믹·리로케이션이 必要하게 된다.

8. 다이나믹·리로케이션

다이나믹·리로케이션은 다음 세가지 方式이 基本으로 되어있다. 어느 것이나 實行時 어드레스의 變換이 이루어지는 特徵이 있다.

1) 베이스·레지스터(base register)에 依한 方法

2) 페이징(paging)에 依한 方法

3) 세그멘테이션(Segmentatin)에 依한 方法

① 베이스·레지스터에 의한 方法

베이스·레지스터에 의한 方法은 가장 簡單하고 잘 應用되는 方法이다. 그리고 다른 두가지 方法의 基礎도 된다. 베이스·레지스터는 그 레지스터의 內容을 命令의 어드레스部에 加하여 그 結果를 휘지칼란 어드레스로서 命令이 實行되므로서 소용되는 기능을 담당한다. 그리고 코아에 프로그램이 로오딩될때 베이스·레지스터에 格納開始番地를 주어, 간혹 로오딩때에 다른 위치에 再配置되어도 命令實行時點에서는 베이스·레지스터에 의해서 휘지칼·어드레스가 變更된다. 即 로지칼·어드레스空間에서 휘지칼·어드레스空間으로의 寫像은 베이스·레지스터에 값을 주므로써 이루어진다. 그리고 소프트웨이에서는

1) 現在 로오드하려는 프로그램이 連結된 메모리의 빈 쁠럭의 하나에 꼭 맞게 들어가는가?

2) 큰 連續된 쁠럭을 만들려면 기왕에 들어가 있는 순서중에서 별로 소용되지 않는 것중 어느 것을 바꾸어 내는 것이 適當한가?

3) 메모리에 프로그램을 되로록 가득 채울 것

등의 일을 하고 있다. 이것이 시스템·오버헤드가 되어, 이 方法에서는 프로그램을 위해서 반드시 연속된 코어領域을 準備하지 않으면 안된다.

② 페이징에 의한 경우

하드웨어上에서의 命令의 어드레스部를, 페이지番地를 나타내는 部分과 페이지內에서의 라인番號를 나타내는 部分으로 나눈다. 페이지의 크기는 시스템에 의해 다르나, 적은 경우는 64~256語정도, 큰 경우는 512~2,048이 취해지고 있다. 이것은 코아메모리를 휘지칼한 페이지로 分割한 것으로, 그 각각은 어느 프로그램 또는 데이타에 對하여 實行段階에 로오드되는 하나의 페이지로 使用된다. 페이지로 分割하여 코아上에 配置되는 프로그램 및 데이타는, 반드시 連續된 페이지에 割當될 必要는 없다. 슈파바이저가 가지고 있는 페이지·테이블에 의하여 이들의 연관이 유지된다. 그리고 命令의 實行에 있어서 로지칼·어드레스에서 휘지칼·어드레스로 轉換이 이루어진다. 따라서 스와프를 해서 새 프로그램을 가져왔을 때에는 페이지·테이블은 更新되어야 한다. 그림 1에 로지칼·어드레스에서 휘지칼·어드레스로의 轉換의 一例를 표시 하였다.

③ 세그멘테이션에 의한 경우

세그멘트란 一般的으로 프로그램의 構成하는 메인·루우틴, 서브·루우틴의 하나 하나, 데이터의 몇個의 집합등의 각각이 對象이 된다. 세그

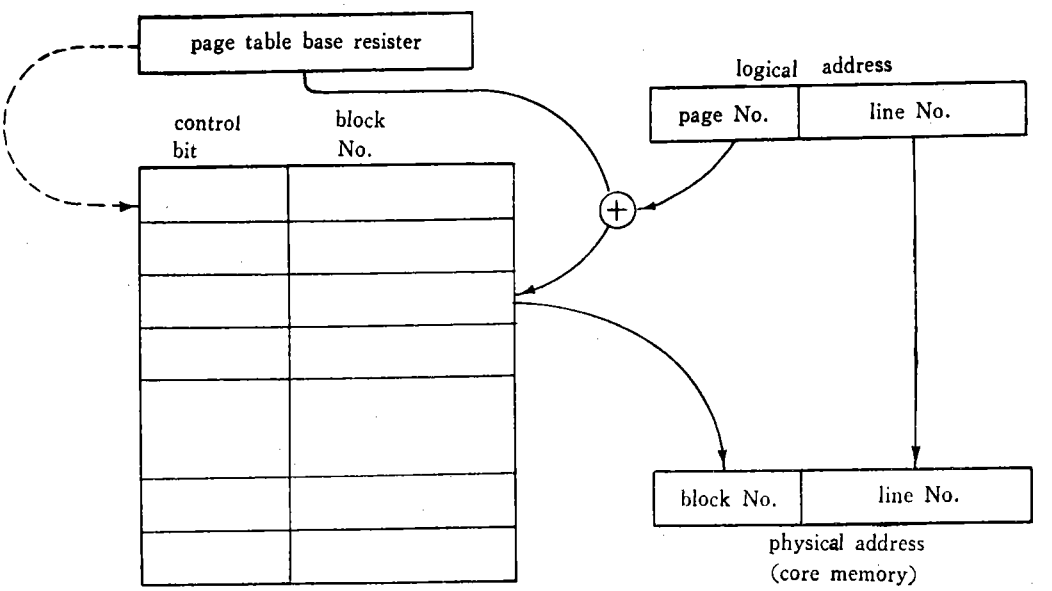

그림 1. 로지칼·어드레스에서 휘지칼·어드레스에의 전환

멘트에는 記號로된 이름이 붙여지고, 그 세그멘트명으로 引用되는 세그멘트內의 하나의 데이타의 어드 레스는, 기호로된 요소 이름으로 표시된다. 例컨대 〈S〉/[α]로 어드레스部가 構成된다면, 〈S〉는 S라고 이름붙여진 세그멘트를 나타내고, [α]는 α라고 이름 붙여진 그 세그멘트 S內에서의 記號로된 요소를 意味한다. 〈S〉는 로지칼·어드레스로서의 세그멘트番號로 번역되고, [α]는 세그멘트內의 相對어드레스로 轉換된다. 세그멘테이션은 로지칼·어드레스空間內의 어느 特定의 요소가 두개의 이름의 組로 參照되므로, 때때로 2次元 로지칼·어드레스空間으로 取扱된다.

페이징에서의 페이지와 라인番號는 user 쪽에서는 보이지 않으나, 세그멘트化된 시스템에서는, user의 프로그램에서, 例컨대 아셈블러에서 〈S〉/[α]의 어드레스를 쓸수 있다.

슈퍼봐이저에서는 테이블레지스터, 세그멘트·테이블, 페이지·테이블을 準備하고 있어. 로지칼·어드레스의 세그멘트番號 n에 테이블·레지스터의 內容이 첨가되어, 이 프로그램에 對한 세그멘트·테이블이 참조된다. 그리고 세그멘트·테이블의 n番째가 指示되어 이 테이블위에서 다음에 對應하는 페이지·테이블의 位置가 얻어져, 여기에 로지칼·어드레스 P가 첨가되고, 페이지·테이블의 m번째가 指示되어, 必要한 페이지의 格納開始番地가 정해진다. 마지막으로 라인·번호 l이 첨가되어 휘지칼·어드레스가 된다(그림2 참조). 위에 말한 轉換過程은 로지칼·어드레스를 參照할 때 必要하게 되어, 그때마다 세그멘트·테이블과 페이지·테이블의 두개의 여분의 코아에 대한 액세스가 따르게 된다. 이것은 프로세서의 能率을 크게 低下시키게 된다. 그러나 이것은 다소 하아드웨어를 複雜化함으로서 除去할 수 있다. 그 일예는 애소시에이티브(associative)메모

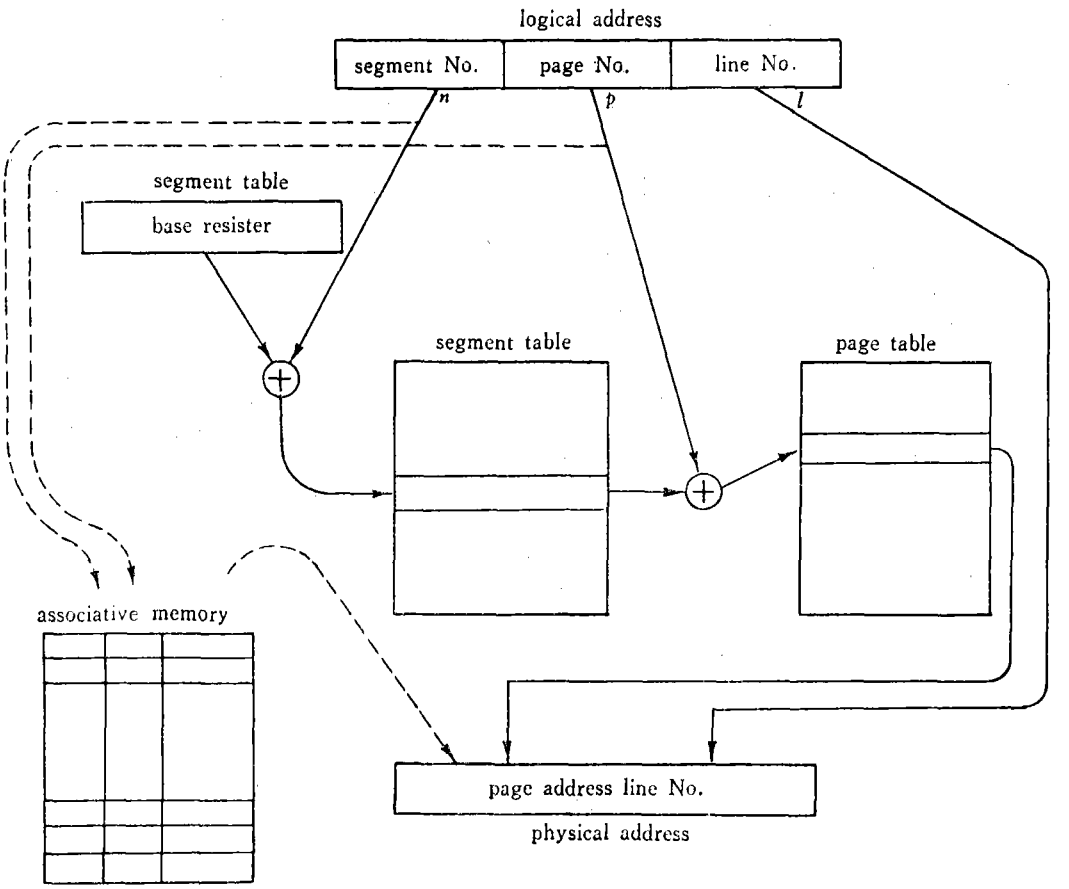

그림 2. 로지컬에서 휘지컬에의 전환에.

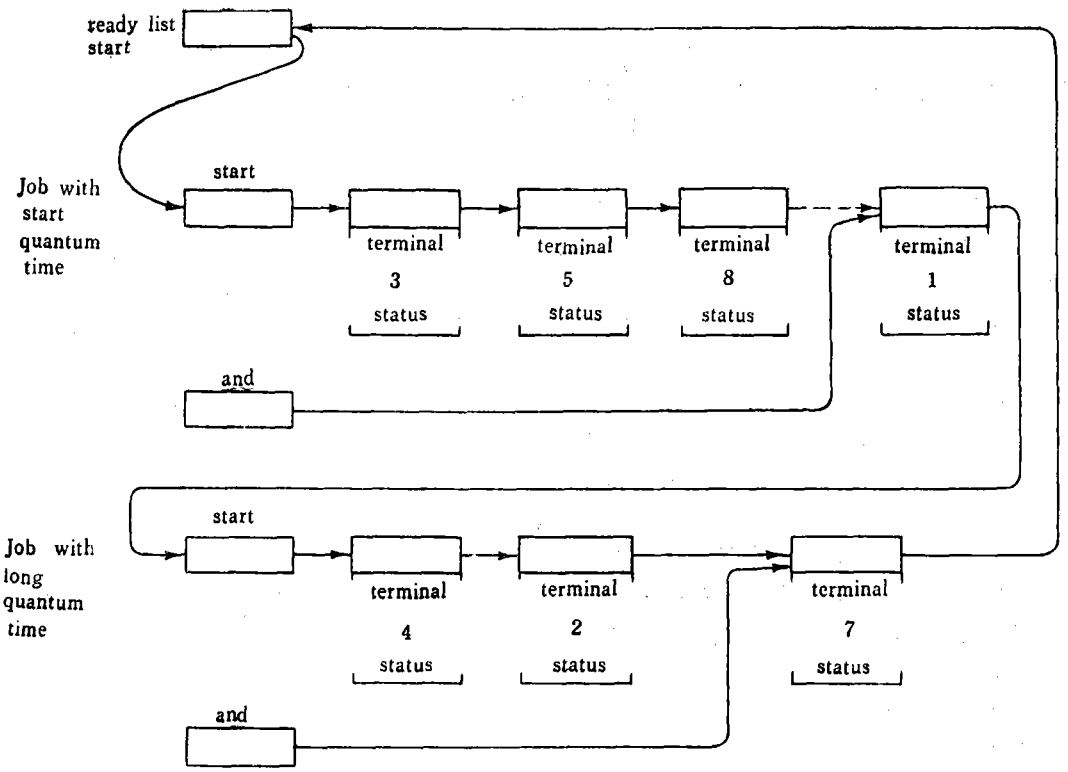

그림 3. 스케줄링의 예

리의 적용이다.

9. 스케줄링 · 알고리즘

타임쉐어링 · 시스템에서 중요한 것은 同時에 액세스하는 多數의 user의 일에 對하여 어떤 순서로 어떤 機能을 割當하여, 어느만큼의 CPU 타임을 주어 일을 처리하느냐 하는 것을, 통활하여 관리하는 소프트웨어로서의 스케줄러이다.

그중 각 일의 實行順序를 管理하기 위한 레디 · 리스트(Ready list)에 대해서 약술한다. 많은 일중 다음에 어느 일에 대한 것을 實行하느냐는 것은, 각각의 일에 관한 情報의 리스트를 管理하므로써 이루어진다. 이것을 그림 3에 표시한 바와 같이, 프로그램을 처리하는 順序는 리스트構造로 順序지어져 있으며, 스타트 · 셀이 指示하는 일부터 차례로 리소오스(resource)라고 총칭되는 quantum · 타임, 메모리領域, 스와핑을 위한 루우틴, 로오다등의 쏘오스가 配當되어간다. 그리고 가장 簡單한 스케줄링의 알고리슴은 FIFO 리스트를 構成하는 라운드 · 로빈(round robin)이라고 부르는 스케줄링이다. 또 일의 內容에 따라 優先度를 고려한 例를 그림 3에 표시하였다.

10. 파일 · 시스템

quantum · 타임의 配當을 中心으로 한 CPU의 分割使用만으로는, TSS의 應用範圍는 매우 좁아진다. 특히 大規模의 情報처리 시스템에서는 파일 · 시스템은 本質的인 존재이다. 一般的으로 要求되는 機能을 들어 보면 다음과 같다.

① 使用者가 적당한 기억하기 쉬운 이름으로 액세스할수 있고, 파일의 작성, 一部의 變更, 削除가 簡單하게 된다.

② 파일된 情報가 多數 user의 일에 의한 共用과, 개인의 特定파일의 機密維持의 確立.

③ 파일에 액세스화는 클래스(리이드 온리, 리이드, 라이트, 자유, 실행 등)의 制御可能.

④ 使用者가 應用時 便利한 파일構造인 점.

⑤ 파일을 通하여 會話를 할 수 있다는 점.

⑥ 事故時에 對備해서, 백·업되는것.
등이다.

파일의 共用과 機密保護의 矛盾된 要求에 對해서는 對象이 되는 파일, 또는 user에게 패스워드를 붙이므로서 키이의 역활을 하게 할수 있다. 그러나 事故等의 非常時에 對備하여, 정기적인 情報의 保存 機密維持等에 대해서 user도 어느 정도 配慮해야 할 것이다.

11. TSS와 使用言語

言語시스템에 관해서는 여기에서는 깊이 다루지 않고 있다. 筆者의 見解로는 TSS에서의 프로그램 debugg도 물론 중요한 일의 하나이지만, 시스템을 提供하는 側에서 大部分의 user의 일에 대처할 수있는 應用프로그램의 蓄積이 이루어졌을 때에는, 言語는 그 專門分野에 관련된 簡單한 問題用의 말로 일을 制御하고, 데이터를 주는 것만으로 情報處理를 할수 있으리라는 理由때문이기도 하다. 따라서 이를 다루자면 그것만으로 相當한 內容을 가진 것이 될 것이다.

TSS에는 시스템에 사용하기 위한 言語로서 코맨드·시스템을 가지고 있다. 코맨드·시스템에는 다음과 같은 機能을 가진 言語가 相當數 準備되어 있다.

① 시스템과의 會話의 開始 終了를 행하는 코멘드

② 파일의 內容의 作成, 變更, 出力을 행하는 코멘드

③ 파일의 이름의 變更, 액세스·클래스의 變更等, 파일의 디렉토리를 制御하는 일을 위한 코멘드

④ 콤파일러를 選擇하던지 프로그램의 實行을 콘트롤하는 코멘드

⑤ 파일이나 시스템에 關한 情報의 索引, 문의를 위한 코멘드
등이다.

12. 結 論

以上 타임쉐어링·시스템의 몇가지 問題에 관하여 말했으나, 콤퓨터·유티리티가 一般的으로 電力과같이 普及되고 大規模情報시스템의 存在下에 社會가 運營될 때에는, 마치 電力事情에 있어서 저녁 등에 일어나는 急激한 電力需要의 增加와 같이, 어느 時間帶에 일어나는 콤퓨터파워의 消費의 피이크를 어떻게 카바하느냐 또는 障害時에 있어서 情報交換은 어떻게 할것인가등의 시스템을 생각할때, 많은 문제 점을 동반한다. 콤퓨터시스템도 大規模情報處理시스템으로서 견딜수 있는 하아드웨어와 소프트웨어가 調和된 發展을 하여야 되리라고 본다.