Journal of Korea Multimedia Society Vol. 14, No. 3, March 2011(pp. 449-461)

CPU 스케줄링을 학습하는 운영체제 시뮬레이션 프로그램의 설계 및 구현

정성균[†], 이상곤[††]

요 약

컴퓨터 운영체제는 과거에는 대학에서만 배우는 과목이었으나, 컴퓨터가 점점 보편화되면서 고등학교와 중학교에서도 운영체제의 교육이 실시되고 있다. 학교에서 이루어지는 컴퓨터 교육을 살펴보면 컴퓨터에 대한 기본 원리와 핵심 철학은 이론 수업만으로 진행되고 있다. 이론 수업은 학습 매체의 활용 부족으로 때로 중요한 부분을 지나치거나 학습자가 흥미를 느끼지 못해 수업이 자칫 어려운 공부로 인식될 위험이 있다. 이러한 교육 환경에서 학습자에게 이론 수업에 추가로 다양한 매체를 활용해서 학습에 대한 이해를 돕고 수업 흥미를 유도해서 과목에 대한 이해를 높이는 연구가 필요한 실정이다. 본 논문에서는 프로그램을 구현하여 운영체제의 이론 중 CPU 스케줄링에 대해서 다양한 예를 진행 과정과 함께 시뮬레이션 하고 그 결과를 표시하는 프로그램을 개발하였다. 본 연구를 통해 개발된 프로그램을 이용하여 학습자를 교육하면 수업에 대한 흥미를 높일 수 있고 컴퓨터 교과의 이론적인 부분을 보강할 수 있다. 또한 다양한 프로그래밍 환경에서의 각 알고리즘(FCFS, SJFS, PS, RR)끼리의 비교 분석도 가능함을 입증한다.

Design and Implementation of Simulation Program for CPU Scheduling in Operating Systems

Seong-Kyun Jeong[†], Samuel Sangkon Lee[††]

ABSTRACT

In the field of computer science, operating system concept is taught in university, but we now teach it in the middle and/or high school. Computer is also taught not only in college but also in middle and high school. If we look up the education of computer that is trained in school, basic principles or core techniques of computer science is educated only with its theory. If the theoretical education of computer science is just trained, sometimes students are not interested in it because of lack of shortage of mass media. Therefore, we could say that it is important that the computer education features a diverse range of media, including prints, paintings, sculpture, digital photographs, mixed media, and a simulation program. For all this reason, we design and implement a program for simulation with computer operating systems especially, CPU scheduling. There are many CPU scheduling algorithms we suggest to make students understand scheduling with some different examples in practical use. In this paper, we practically propose a new approach to be used with a study tool to make a motivation for students. We design a simulation program for teaching computer operation systems to show CPU scheduling and we implement a program to make use of comparison of FCFS, SJFS, PS, and RR scheduling algorithms. With our simulation program we present a comparative analysis between scheduling algorithms could be possible.

Key words: Computer Education(컴퓨터 교육), Simulation(시뮬레이션), CPU Scheduling(CPU 스케줄링), Operating Systems(운영체제)

※ 교신저자(Corresponding Author) : 이상곤, 주소 : 전라북도 전주시 완산구 천잠로 303번지(560-759), 전화 : 063)220-2934, 팩스 : 063)220-2056, E-mail : samuel@jj.ac.kr
접수일 : 2010년 8월 24일, 수정일 : 2010년 12월 13일
완료일 : 2011년 1월 17일
[†] 정회원, 전라북도교통문화연수원
(E-mail : traxez@nate.com)
[††] 종신회원, 전주대학교 컴퓨터공학과 부교수

1. 서 론

최초의 컴퓨터가 등장한 지 불과 60년 남짓한 시간이 흘렀음에도 어제의 진리가 오늘에 와서는 그 의미를 상실하게 될 만큼 빠른 변화를 겪는 학문 분야가 바로 컴퓨터공학 및 정보기술(IT) 분야이다. 하지만 컴퓨터 분야의 기본 원리 및 정보 기술이 추구하는 핵심 철학은 시대가 흘러도 변하지 않고 있다.

현대에는 과학 문명의 발달과 함께 과학기술의 여러 기기로 모든 사람이 편리함과 많은 혜택을 누리게 되었다. 인간이 만든 여러 과학기기 중 컴퓨터기술의 빠른 발전 속도로 인해, 한 가정에 최소 한 대 이상의 컴퓨터를 가지게 되었고, 사람들에게 컴퓨터는 다양한 문화의 혜택과 생활의 질을 높이는 하나의 道具(도구)로서 자리 잡게 되었다. 이러한 컴퓨터의 발달로 점차 컴퓨터의 사용 방법이 널리 알려지게 되었고, 그 기기의 이해도가 보편화 되었다. 처음에는 대학에서만 가르치던 컴퓨터 학문이 이제는 고등학교, 중학교, 지금은 초등학교 방과 후 수업에 이르기까지 확대되었다.

학교에서 컴퓨터를 가르칠 때 가장 기본이 되는 하드웨어(H/W, Hardware)와 소프트웨어(S/W, Software)의 개념과 하드웨어와 소프트웨어를 연결하는 운용체계인 운영체제(OS; Operating Systems)의 개념[1]을 정확하게 이해시켜야 학생들에게 컴퓨터에 대해 제대로 가르칠 수 있다. 운영체제는 사용자에게 하드웨어의 제어를 효율적으로 사용할 수 있게 도와주고 동시에 소프트웨어를 효과적으로 사용할 수 있도록 하는 기반이 되는 가교 역할을 한다. 그리고 운영체제의 동작은 사용자에게는 최대한의 편리함을 제공하여, 컴퓨터를 효과적으로 사용하도록 구성되어 있다.

실제로 이러한 컴퓨터 교육의 운영체제의 교과 내용은 다양한 형태의 H/W에서부터 사용자들에게 친근한 S/W까지 그 활용 방안이 확대되어 가고 있으며, 각종 IT 관련 자격증, 학술대회, 연구회, S/W 경진대회를 통해 널리 활용되고 있다. 이와 같이 컴퓨터 교육이 교육 현장에서 널리 교수되고 있음에도 불구하고, 현재 컴퓨터 교육의 주된 내용은 교과서 위주의 이론 수업으로만 진행되고 있어, 학생들에게 수업의 이해를 돕기 위한 다양한 형태의 매체 사용이 절실하다. 교육 매체의 다양성이 부족하다는 문제는 컴퓨터의 기본인 하드웨어, 소프트웨어, 운영체제를 이해하지 못한 체 넘어가게 되며, 이것은 학생들의 집중력 저하로 자칫 흥미 있는 컴퓨터 교과가 지루한 과목이 되어 버린다. 이러한 문제점을 보완하기 위해 ICT; Information and Communication Technology 활용 수업이나 각종 사이버 학습, 파워포인트를 활용한 수업, 기타 이외의 다른 응용 프로그램을 사용하여 이론수업 위주의 지루함을 없애 주는 방안이 연구되어 있다[2,3].

컴퓨터 운영체제는 하드웨어를 효과적으로 사용할 수 있도록 하며, 소프트웨어가 구동될 수 있는 기반을 제공하도록 발전되어 왔다. 즉, 운영체제는 다양한 소프트웨어들이 하드웨어를 잘 운용할 수 있는 기반을 마련해 주는 것과 같은 의미인데, 소프트웨어가 하드웨어를 활용함에 있어서 자원, 프로세스, 기억장치 등 컴퓨터의 주요한 구성요소들 사이의 유기적인 정보를 빠르게 파악하여 상보적으로 구동될 수 있도록 도와주는 역할을 하고 있다.

컴퓨터의 운영체제가 하나의 일만을 처리한다고 하면 프로세스 상태나 메모리 상태를 점검할 필요 없이 현재 실행되는 하나의 작업이 끝나기를 기다렸다가 다음 작업을 처리하면 되지만, 현대의 복잡한 컴퓨터 시스템은 멀티태스킹(다중 프로그래밍) 시스템으로 한 번에 하나 이상의 여러 프로세스(process)가 독립적으로 운용되는 컴퓨터 구조(computer architecture)를 가지고 있다. 이러한 멀티태스킹 개념으로 인해 운영체제는 더욱 더 효과적으로 소프트웨어를 관리하도록 발전해 가고 이러한 처리 방법을 "CPU 스케줄링(CPU Scheduling)"을 통해 최적의 방안으로 운용될 수 있도록 설계되어 있다.

본 논문에서는 운영체제의 여러 가지 주요 임무 중 H/W와 S/W가 가장 밀접한 관계를 정의하고 있는 알고리즘의 하나인 CPU 스케줄링에 대해 실제 컴퓨터 운영 환경에서 시뮬레이션 되는 과정을 설계하여 구현함으로서 교육 수요자들에게 컴퓨터 운영체제의 동작 방법에 대한 올바른 이해와 흥미를 유도하는데 목적이 있다. 또한 CPU 스케줄링의 시뮬레이션 동작을 시각적으로 보여 줌으로써 수업 집중도의 향상과 기타 부가적인 효과(프로그래밍적인 요소 혹은 각종 매체의 사용)를 꾀하고자 한다[4]. 본론에 앞서, 앞에서 언급한 내용을 토대로 기본적인 이론 수업에 컴퓨터 응용 프로그램을 접목시켜 학생들에게

이해력을 돕고, 수업 시간의 집중력을 높이며, 이론 위주의 과목을 실습과 병행해 지도하여 학생들에게 컴퓨터 교과의 흥미를 유도하고자 한다[3].

본 논문에서는 독자가 운영체제의 여러 스케줄링 알고리즘에 대한 기본적인 이론적 내용은 알고 있다고 가정하고, 스케줄링 알고리즘을 실제 프로그램에서 어떻게 구현하였는지 설명한다. 또한 학생들이 서로 다른 알고리즘들의 특징을 고려하여 직접 동작해 봄으로서 운영체제의 CPU 스케줄링 방법을 이해하여 이론적이고 실제적인 내용을 자기의 것으로 만들도록 한다. 이러한 접근 방법은 교수 학습의 일부분으로서 학생들에게 이론 수업을 진행한 후, 실습해 보는 시간을 갖도록 지도하여 배운 교과 내용을 확실하게 이해하였는지 점검한다. 또한 프로그램의 動機化(동기화)를 통해 컴퓨터 프로그램의 작동 원리에 대해 탐구할 수 있으며, 동시에 학습자가 실제로 수행되는 프로그램의 처리 순서와 해당 알고리즘에 대한 설정값을 직접 입력하고 그 변화 과정을 살펴봄으로서 컴퓨터 운영제체와 컴퓨터 프로그래밍의 학습 효과를 극대화하도록 하였다[2,3].

본 논문의 구성은 다음과 같다. 제2장에서는 본 논문과 관련된 이론적인 연구인 운영체제와 스케줄러에 대해 살펴보고, 3장에서는 본 논문에서 제안하는 운영체제의 프로세스와 자원 관리(Process and Resource Management)를 체계적으로 관리하는 시스템에 대한 개요와 이에 따른 알고리즘, 그리고 구현 기술에 대해 논의한다. 제4장에서는 구현된 운영체제의 프로세스와 자원 관리 프로그램을 소개하고, 실험 결과에 대해 간략히 설명한다. 마지막으로 5장에서는 결론과 향후 연구 과제에 대해 기술하였다.

2. 관련 연구 및 설계 요구 사항

2.1 현 교육 실태의 문제점 진단

컴퓨터의 가장 기본이 되는 하드웨어와 이 하드웨어를 효율적으로 운용할 수 있도록 설계된 운영체제는 컴퓨터의 기본적인 학습 내용임에도 불구하고, 이를 교육하기 위한 여러 도구의 제작에 대한 노력은 여전히 부족한 실정이다. 컴퓨터 운영체제의 동작 과정은 실제로 학생들의 눈으로 보이지 않아 그 처리 과정을 일일이 확인할 수가 없으며, 운영체제를 이해하기 위해 하드웨어와 소프트웨어를 관리하고 구동하려면 알고리즘별로 서로 다른 적당한 운영 환경을 제공해 주어야 하기 때문에, 컴퓨터 공학 기술에서도 상당히 어려운 분야에 속한다.

이에 따라 운영체제 교과에 대한 여러 교육 방법이 제시되는데, 브루너의 교수 이론 방법[2]에 따르면 "학습자에게 제공되는 교수는 직접적인 경험에서부터 그림이나 필름과 같은 영상적인 표현과 그 다음으로 언어와 같은 상징적인 표현의 순서로 제시되어야 한다고 제안하고 있다. 직접적 경험이나 영상적인 표현은 언어와 같은 상징물보다는 의미의 전달 면에서 더 정확하고 이해하기 쉽다"고 설명하였다. 언어는 상당히 추상적인 형태이기 때문에 학습자에게 언어만 제공되는 경우에는 학습자은 어려움을 느낄 수 있기 때문이다. 따라서 추상적인 개념을 제시할 때에는 글이나 언어는 물론 그래프나 그림과 같은 영상 매체를 이용하는 방법이 교육 수요자의 이해를 촉진할 수 있게 된다. 교수 매체는 이러한 구체적인 경험을 제공하는데 중요한 역할을 할 수 있다. 본 논문에서 구현한 프로그램을 사용하는 교수 매체는 도표나 그림 혹은 움직임을 이용하여 이론과 유사한 경험을 학습자에게 제공하므로 교육의 이해를 극대화 할 수 있다[2]. 이러한 부분은 실제 수업 현장에서도 확인할 수 있는데, 수업 시간에 이론적인 부분만 설명하는 것보다는 파워포인트를 활용한 발표 수업인 경우에 학생들의 집중도가 높이 나타난다고 알려져 있으며, 특정 내용을 수업할 때 그 이론 내용과 관련된 실습이 추가적으로 더해지면, 학생들의 이해력은 훨씬 높게 나타나게 된다[3]. 실제로 수업 집중도에 있어서도 이론 수업인 경우 학생들이 시간의 흐름에 따라 지루함을 느끼는 반면, 이론과 실습을 병행한 수업 혹은 실습 수업의 경우에는 시간의 경과하여도 계속해서 흥미를 느끼며 집중도가 유지되는 것으로 보고되고 있다[4]. 위와 같은 사실을 근거로 본 논문에서는 운영체제의 프로그램을 개발하여 학습자의 흥미 위주로 교육이 가능하도록 한다는 목표로 본 논문의 논의를 진행하고자 한다.

2.2 운영체제 교과의 지도

운영체제(Operating Systems)란 컴퓨터의 하드웨어를 효율적으로 관리하는 프로그램이며, 응용 프로그램을 위한 기반을 제공하며, 사용자에게 하드웨어와 소프트웨어의 중재자 역할을 수행한다[1]. 컴퓨

터 시스템은 크게 하드웨어, 운영체제, 응용 프로그램, 사용자 등으로 분류하며, 운영체제는 하드웨어의 주요 장치들인 중앙처리장치, 메모리, 입/출력 장치 등의 자원을 관리하고 있으며, 우리가 자주 사용하고 있는 한글 워드프로세스 프로그램, 마이크로소프트사의 MS 오피스 프로그램 등 각종 응용 프로그램이 하드웨어를 효과적으로 구동하도록 자원을 관리하며 제어하는 역할을 하고 있다. 운영체제는 자원 관리자(Resource Manager)의 관점 이외에도 여러 다양한 방법으로 관리되며, 사용자에게 최적의 환경을 제공하는 역할을 한다.

2.3 CPU 스케줄링의 다양한 모델

CPU 스케줄링은 다중 프로그래밍을 지향하는 운영체제의 가장 기본이 되는 핵심 부분이다. CPU의 이용률을 최대로 가용하며 항상 실행중인 프로세스(process)를 가지게 하며, 여러 형태의 효율성과 생산성을 고려하여 운용되어야 한다.

스케줄링은 하나의 프로세스가 실행 중에 입출력 상태에 도달하면 CPU는 입/출력 요청이 끝날 때까지 그동안 수행하였던 프로세스를 더 이상 진행하지 못하게 된다. 이러한 지연(delay) 시간을 극복하기 위해 입출력 대기 중인 현재의 프로세스를 종료하고 다른 프로세스를 CPU에게 할당한다. 이와 같은 처리 방법에 있어 CPU는 다양한 환경에서 다양한 조건으로 프로그램의 우선순위를 정하고 활성화 되는데, 이러한 처리 방법을 스케줄링(scheduling)이라고 한다[1]. 다음 절에서는 CPU가 사용자의 여러 작업(job)을 스케줄링 하는 여러 방법들을 비교 설명하기 위해 참고 문헌 [1]에서 예시한 표를 이용하여 설명하고자 한다.

2.3.1 FCFS

선입 선처리(First Come, First Served) 스케줄링은 가장 간단한 CPU 알고리즘 중 하나이다. 이 알고리즘의 구현은 선입 선출 큐(queue)로 쉽게 구현이 가능한데, CPU가 자유 상태(free)가 되면, 준비 완료 큐의 앞 부분에 대기하고 있던 프로세스에게 CPU를 할당하고 곧바로 실행된다. 처음으로 큐에 도착한 프로세스의 작업이 완료되기 전에는 CPU는 결코 자유로운 상태가 되지 못하며, 반드시 처음 프로세스의 작업이 완료된 후에야 다음 프로세스로 그 제어를 이동하는 구조이다. 이러한 스케줄러의 구현은 추가적인 절차 없이 쉽게 구현 가능하다. 그러나 이 스케줄러의 구조는 큐에 처음 들어 온 프로세스가 오랫동안 실행되면 다음 프로세스의 실행 시간과 상관없이 모든 프로세스가 CPU를 기다리게 된다는 단점이 있으며, 그와 함께 다른 프로세스들의 평균 대기 시간도 길어지게 된다[1].

표 1. 프로세스와 CPU 버스트 시간의 예 ①

프로세스	버스트 시간
P1	24
P2	3
P3	3

위의 표 1에 제시된 바와 같이 세 개의 프로세스가 P1, P2, P3의 순서로 작업큐에 도착하여 CPU의 서비스를 기다린다고 가정해 보자. 표에서와 같이 실행을 마친 후 프로세스 P1의 대기 시간은 '0', P2는 '24', P3는 '27'이 되므로, 각 프로세스들의 평균 대기 시간은 $\frac{0+24+27}{3} = 17$밀리초(ms)로 계산된다. 그러나 프로세스의 순서가 표 1과는 다르게 도착하면 예를 들어 P3, P1, P2로 도착하면 평균 대기 시간은 $\frac{0+3+27}{3} = 10$밀리초(ms)가 되기 때문에 평균 대기 시간이 앞의 경우에 비해 현저히 감소한다. 따라서 FCFS 알고리즘은 대기열에 도착하는 프로세스의 순서에 의해 컴퓨터 성능에 큰 영향을 준다[5].

2.3.2 SJFS

최단 작업 우선(Shortest-Job-First) 스케줄링은 각 프로세스에 다음 CPU 버스트(실행되어야 하는 시간)의 길이와 연관하여 수행된다. CPU의 이용이 가능해지면 가장 작은 CPU 버스트를 가진 프로세스에게 CPU를 먼저 할당한다. 이 스케줄링 방식은 작은 프로세스가 먼저 처리되기 때문에 일반적으로 최소의 평균 대기 시간을 가진다는 점에 있어서 스케줄링의 성능이 최적이다. 그러나 다음 요청시 CPU 버스트를 정확하게 예측할 수 없다는 문제점이 있다. SJF의 종류에는 비선점형과 선점형으로 나눌 수 있는데, 선점형의 경우 계속 CPU 버스트가 적은 프로세스가 나타나게 되면 버스트가 큰 프로세스의 처리가 무기한 연기될 수 있다. 비선점형인 경우 실시간 시스템에서 버스트가 큰 프로세스가 먼저 도착하면

표 2. 프로세스와 CPU 버스트 시간의 예 ②

프로세스	도착 시간	버스트 시간
P1	0	5
P2	1	2
P3	2	3

그 프로세스를 처리하는데 많은 시간 지연이 발생할 수 있다.

위의 표 2에서와 같이 가장 먼저 도착하는 프로세스는 P1이며, 이 프로세스가 처리되는 동안에 P2가 도착하여도 P1이 수행을 종료하려면 아직 4밀리 초가 남아 있기 때문에 계속하여 P2가 선점되어 처리된다. P2가 끝나는 시점인 3 밀리초 후에는 P3도 이미 도착해 있기 때문에 P1보다 남아 있는 버스트 시간이 상대적으로 적은 P3가 CPU를 할당 받으며, 마지막으로 P1 프로세서가 실행된다. 이에 따라 평균 대기 시간은 $\frac{(6-1)+(1-1)+(3-2)}{3} = 2$밀리초(ms)가 된다. 이전의 2.3.1절의 FCFS에 비해 평균 대기 시간이 현저히 감소한 것을 알 수 있어 SJFS가 훨씬 효율적인 알고리즘이다[6].

2.3.3 PS

우선순위(Priority) 스케줄링은 각 프로세스의 수행에 우선순위를 연관하여 처리하는 방식이다. CPU는 가장 높은 우선순위를 가진 프로세스에게 할당된다. 만약 우선순위가 동일한 경우는 선입 선처리로 스케줄링 된다. 여기서는 프로세스의 버스트 시간과 관계없이 우선순위만을 확인해 실행되기 때문에 우선순위 이외에 다른 조건은 무시된다. 이와 같이 우선순위 스케줄링에서 나타나는 주요 문제는 무한 봉쇄 또는 기아 상태(starvation)이다. 실행 준비는 이미 되었으나 우선순위가 낮아 CPU를 사용하지 못하는 프로세스는 무한정 대기하는 경우가 발생할 수 있다. 이러한 문제점을 해결하기 위한 방법으로 노화(aging)라는 새로운 개념이 나왔는데, 노화란 우선순위에 밀려 오랫동안 실행이 되지 못하는 프로세스를 탐지하여 이 프로세스에게 보다 높은 우선순위를 부여하여 시간의 흐름에 따라 점진적으로 우선순위를 높여 주는 구현 기술이다.

2.3.4 RR

RR(Round-Robin) 스케줄링 기법은 고정 크기 퀀텀 q 또는 타임 슬라이스(time slice) 동안 프로세스가 연속적인 CPU 시간을 점유하도록 한다. 프로세스가 연속적인 시간 단위인 q를 모두 사용하면 그 프로세서는 CPU에게 선점 당하고 프로세스를 기다리는 프로세스 리스트의 제일 끝(뒷부분)에 들어가게 된다. 모든 프로세스는 같은 우선순위를 가지고 있기 때문에 순환 중재 규칙이 적용된다.

3. 프로세스와 자원 관리용 시뮬레이션 프로그램의 설계

3.1. 시뮬레이션의 개요

운영체제의 스케줄러는 많은 종류의 알고리즘이 학계에 보고되어 있으며, 수많은 교재와 논문이 존재하지만 여전히 교육 수요자가 이해하기에는 어려운 부분이 있다. 또한 실제 스케줄링 알고리즘이 어떻게 구현되어 있으며, 어떻게 실행되는지, 각 알고리즘들끼리의 비교는 어떤지 학생들이 이해하는 것이 쉽지 않다. 따라서 본 논문에서는 주요 알고리즘의 구현 기술에 대해 소개하고 이를 실습해 볼 수 있으며, 후속 연구자들이 본 논문의 구현 기술을 이용할 수 있도록 하고자 한다[7].

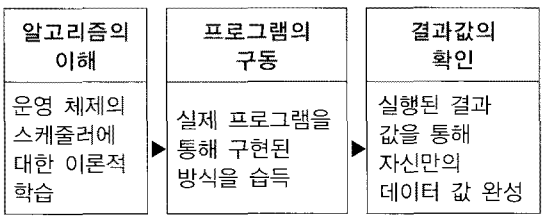

그림 1. 프로그램의 전체적인 실행 과정 및 구조 설명

본 논문에서는 위의 그림 1과 같이 사용자가 알고리즘을 이론적으로 이해하고, 프로그램을 실제적으로 구동하여 보고, 그 결과값을 확인할 수 있는 프로그램의 다양한 구동 환경을 제공한다. 구동 환경은 윈도우즈 환경에서 구동이 되도록 하며, GUI 방식을 통해 프로그램에 능숙하지 못한 학습자도 쉽게 구동해 볼 수 있도록 하였으며, 실제 구동된 결과 값들은 데이터베이스를 통해 저장하여 추후에도 동일한 시뮬레이션이 가능하도록 설계하였다. 프로그램의 실행을 원하는 사용자는 알고리즘에 대해 배운 내용을 더 깊이 이해할 수 있으며, 이론 수업을 실습 수업으

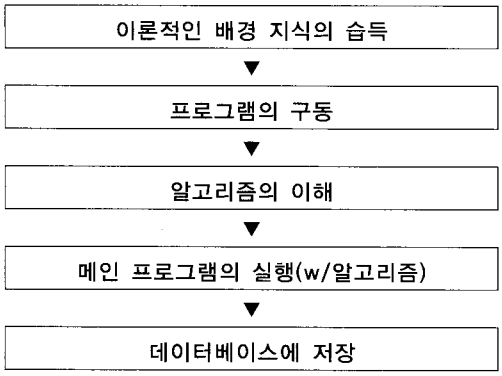

그림 2. 프로그램의 구동 방법

로 전환이 가능하여 학습자의 흥미를 유발하도록 하였다.

위의 그림 2는 사용자들이 실제 프로그램을 구동하는 절차에 대해 설명한다. 이론적인 배경 지식의 습득을 통해 정규 교과에서 배운 내용을 토대로 프로그램을 구동하며, 이론에 대한 설명이 부족한 부분은 프로그램에서 제공하는 삽화를 통해 보충하고, 메인 프로그램을 구동하여 실제 알고리즘이 운영체제 내에서 프로세스와 메모리의 자원 관리가 어떻게 진행되는지에 대해 살펴본다. 다음으로 프로그램 수행값을 데이터베이스에 저장함으로서 추후에도 사용자가 지정한 것과 동일한 실험 결과를 시뮬레이션 해보도록 실습 데이터를 저장한다.

3.2 프로그램의 개발 방향

본 논문의 서두에 언급한 바와 같이, 이론적으로만 수업하는 운영체제의 수업은 자칫 학생들을 지루하게 만들거나 흥미를 저하시킬 수 있다고 지적하였다. 또한 현재 공부하고 있는 내용에 대해서도 확실하게 이해하였는지 실습을 통해 테스트하여 학습 효과를 극대화 할 수 있다. 따라서 학생들에게 여러 CPU 스케줄링 알고리즘에 대한 실습 과정을 시각적으로 제공함으로서 올바른 학습의 이해와 흥미를 유발시키는데 본 프로그램의 구현 목적이 있다. 또한 본 논문에서 제시한 시뮬레이션 프로그램은 학생들에게는 프로그램을 개발하는 동기(motivation)로 작용할 수 있는 장점도 있다.

이미 운영체제의 알고리즘을 이해하고 있는 학생에서는 실습을 통해 이것을 사용하는 도구와 작동원리, 알고리즘에 대해 복습할 수 있는 환경도 제공한다. 개발된 프로그램의 다양한 경험은 추후 업그레이드 될 프로그램의 개발이나 수정 시 참고할 수 있는 주요 재료가 될 것이다.

3.3 프로그램의 구현 기술

3.3.1 준비 도구

이 절에서는 본 논문에서 제안하는 프로그램을 구현하기 위해 사용한 프로그램에 대해 간단히 소개하고자 한다. 프로그램의 구현은 마이크로소프트에서 개발된 C#을 이용하여 개발하였다. C#은 기본적인 프로그램의 작성 도구이며, 동시에 각종 응용 프로그램의 개발이 가능한 언어이다. 이것은 비주얼 C++에서 시작해 2005년부터 C#으로 발전하였다. 본 논문에서 제공하는 프로그램의 실행을 위해 컴퓨터에 .net Framework가 이미 설치되어 있어야 하며, 버전에 따른 호환성 문제가 있을 수 있다[8-10]. 그래픽 작업과 편집을 위해 Photoshop 프로그램을 이용하였다. 이 툴은 미국의 Adobe에서 개발한 그래픽 소프트웨어로 2D 이미지를 제작/편집하는데 중심을 두고 있다. 비트맵 방식을 사용하여 색 처리에는 우수하지만 이미지를 확대하면 계단식 현상이 발생하여 이미지가 깨져 보이는 현상이 발생할 수 있다.

본 논문에서 제공하는 삽화를 제작하기 위해 사용된 모든 움직이는 애니메이션은 Flash 프로그램으로 구성하였다. Macromedia에서 개발된 Flash는 처음에는 애니메이션 위주의 기능을 제공하였지만 웹과 소프트웨어 기술의 융합적인 발달로 인해 Action Script 언어를 통해 몇 가지 프로그램적인 기능들이 추가 되었고, 이제는 하나의 독립된 웹 언어로 자리할 정도로 빠르게 성장한 소프트웨어이다.

3.3.2 사용자 인터페이스

본 논문에서 개발한 프로그램(OSM; Operating System Manager)의 사용자 인터페이스는 사용자의 입장에서 최대한 이해하기 쉽게 구현하였다. 메인 화면은 사용자가 접하는 맨 첫 화면이기 때문에 다른 부분에 비해 첫째, 사용자가 직관적으로 이용할 수 있는 인터페이스 구현과 둘째, 사용자의 학습 요구사항에 따라 필요한 부분을 미리 준비하도록 하였다. 이에 따라 스케치한 도안을 가지고 사용자 인터페이스를 구상한 후, 구상된 자료를 토대로 포토샵 프로그램을 사용하여 메인 화면을 설계하였다[11].

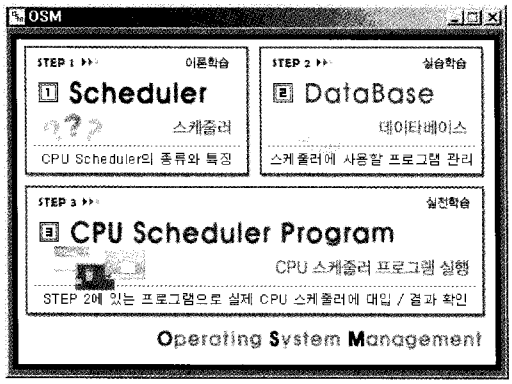

그림 3. OSM의 메인 화면

위의 그림 3과 같이 사용자에게 직관적으로 보이게 하도록 개별 아이콘을 크게 제작하였으며, 각각의 처리 과정에 맞게 이동할 수 있도록 STEP-1, -2, -3로 나누어 배치하였다.

3.3.3 스케줄링 알고리즘의 구현

알고리즘의 구현은 실제 운영체제에서 배우고 있는 기본 알고리즘(FCFS, SJFS, PS, RR 등)을 가지고 최대한 알고리즘의 개별 특성에 맞게 프로그램을 설계하였다. 해당 알고리즘을 프로그램에 반영하기 위해 계산 방법도 중요하지만 이에 필요한 식이나 변수 등에 대해서 프로그램 언어와 동일한지 살펴보고, 실제로 변수에 대입하였을 때 문제가 되는 부분에 대해 미리 검토하여 문제가 발생하지 않도록 변환하는 작업을 하였다. 실제 알고리즘에 대해 분석하고, 프로그램에서 어떻게 활용되는지 확인하여 해당 변수 값에 대입하여 실제적인 테스트와 결과 값의 종합적인 확인이 가능하도록 설계하였다.

```
{
    //고정데이타에 값 추가하기
    vp_pds vp_pds = new vp_pds();
    vp_pds.vp_sid = db_list.GetString(0);
    vp_pds.vp_id = db_list.GetString(1);
    vp_pds.vp_pro = db_list.GetInt32(2);
    vp_pds.vp_atten = db_list.GetInt32(3);
    vp_pds.vp_fcfs = db_list.GetInt32(4);
    vp_pds.vp_mem = db_list.GetInt32(7);
    vp_pds.vp_enter = 0;
    String tmp_string = db_list.GetString(6);
    vp_pds.vp_etime = 0;
    vp_pds.vp_wtime = 0;
    vp_pds.vp_start = i + (list_count_check - 1);

    //리스트 안에 자료 넣기(추후에 문제 생기는것을 방지)
    for (int k = 0; k < vp_pds.vp_pro; k++)
    {
        sch_resource sch_r = new sch_resource();
        sch_r.i_id = vp_pds.vp_id;
        sch_r.i_name = tmp_string;
        sch_r.i_count = (vp_pds.vp_pro - k);
        sch_r.i_fcfs = vp_pds.vp_fcfs;
        if (k == (vp_pds.vp_pro - 1))
        {
            sch_r.i_con = "E";
        }
        sch_r.i_next = list_count_check;
        sch_list.Add(sch_r);
        list_count_check++;
    }
    vp_pds.vp_end = list_count_check - 2;
    v_pds.Add(vp_pds);
    String test_text = db_list.GetString(6);
    wait_image(i, test_text); //이미지 만들기
    i++;
}
```

그림 4. 알고리즘의 변수 매칭

4. 시뮬레이션의 구현 방법

본 시스템의 구현 환경은 메모리 4GB를 탑재한 CPU Pentium D 3.2GHz 속도를 가진 시스템에서 마이크로소프트사의 Visual Studio .Net 2008을 이용하여 구현하였고, Photoshop을 이용하여 이미지 파일을 디자인하고, 데이터베이스는 MS-SQL을 이용하였다. 샘플로 준비해 놓은 데이터를 데이터베이스에 미리 내장하여 교수자가 학습자의 실습 시 곧바로 활용할 수 있도록 미리 계획된 시뮬레이션도 준비하였다. 시스템의 전체 흐름도는 운영체제 교과에서 배운 이론을 가지고 사용자가 충분히 이해했는지 확인한 후, 이해가 되지 않았다면 복습하는 기회를 가지도록 하고, 반면에 이론을 이해한 경우에는 바로 프로그램을 구동할 수 있도록 전체 시스템의 흐름도를 아래의 그림 5에 제시하였다.

처음 프로그램을 이해하는 방법부터 시작하여 프로그램의 구동 방법에 따른 절차에 따라 진행하고, 혹시 이해가 되지 않을 때는 다시 복습할 수 있도록 프로그램을 구성하였다. 인터페이스는 직관적으로 구성하였다. 처음으로 프로그램을 실행할 때는 이론적인 내용을 출력하고, 다음으로 데이터를 입력하는 곳을 따로 구성하였으며, 한 화면에 보이는 기능은 최대한 간단히 처리하였다.

첫 번째 STEP-1에서는 그림 6과 같이 CPU

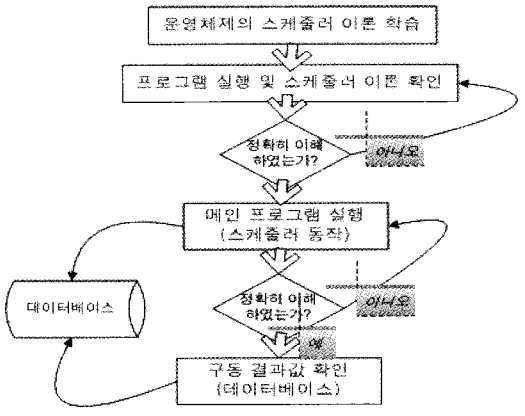

그림 5. 시스템의 전체 구조도

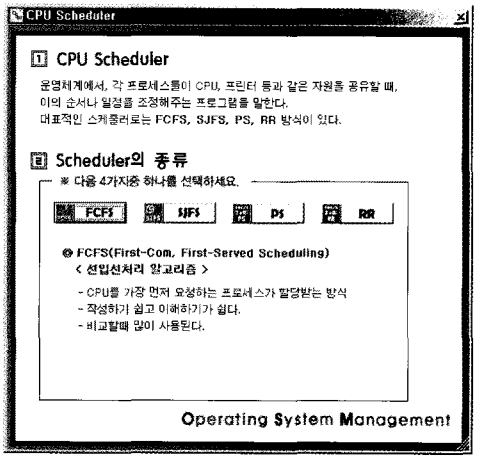

그림 6. CPU Scheduler

Scheduler에 대해 설명하는 부분이다. 여기서는 이론적으로 배운 CPU 스케줄러에 대해 설명하고자 하였고, 프로그램에 구현된 스케줄링 방법을 설명하고 있다.

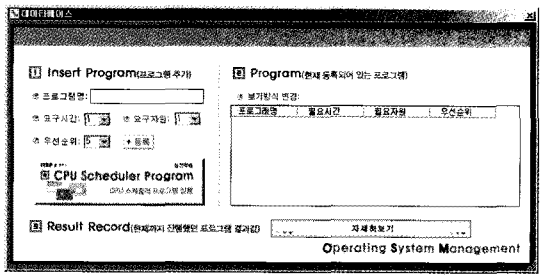

그림 7. 데이터베이스의 제어 화면

두 번째 프로그램에서 그림 7는 현재 데이터베이스에 등록되어 있는 프로그램의 데이터 값을 확인할 수 있도록 하며, 프로세스의 삽입/삭제/수정이 가능하고, 학습자가 지금까지 수행한 여러 상황에 대해 그 결과 값들을 확인할 수 있다. 여기에서 실행의 전환을 빠르게 하기 위해서 바로 CPU Scheduler Program 화면으로 넘어갈 수 있는 화면도 제공하는 기능도 구현하였다.

세 번째 프로그램의 그림 8은 현재 데이터에 입력되어 있는 프로그램을 가지고 실제로 시뮬레이터를 실행하기 위한 준비 작업을 하는 단계로서 프로세스의 설정, 스케줄러의 설정, 프로그램 설정 등의 순서로 구성되어 있다. 프로세스의 수를 선택하고, 스케줄러의 방법을 선택(기본 알고리즘과 RR 알고리즘)

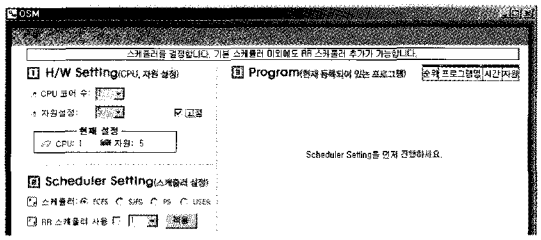

그림 8. 프로그램 설정 화면

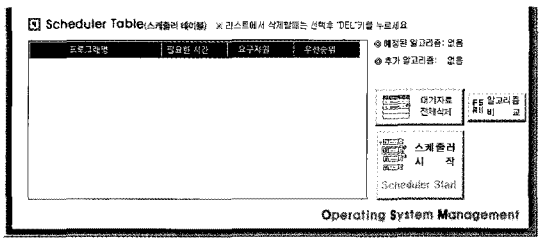

그림 9. 스케줄러 대기 화면

한 후, 마지막으로 실행할 프로그램을 추가하면 설정 작업을 모두 마치게 된다.

모든 기본 설정을 마친 후에는 스케줄러 대기 화면 그림 9에서 현재까지 진행된 부분에 대한 확인/수정이 가능하다. 이 모듈은 총 두 가지로 나누어 있는데, 첫째는 스케줄러의 중간 과정을 보는 "스케줄러의 시작" 부분과 둘째, 스케줄러의 결과 값을 비교해 보는 "알고리즘의 비교" 등으로 구분되어 있다. 사용자는 두 가지 방법 중 자신이 원하는 방법으로 프로그램을 실행한다.

그림 10에서는 선택한 프로그램과 스케줄러를 가지고 프로그램이 어떤 방법으로 진행이 되는지 동적인 형태로 표현하였는데, 단순히 고정적인 이미지가 아닌, 실제 CPU의 위치가 변동되는 방법을 선택하여 동적인 움직임을 보여주도록 설계하였다.

중간에 타이머 기능을 탑재하여 프로세스의 일시정지/재시작이 가능하고, 실행을 종료하는데 남은 시간, 우선순위의 값, 다음 프로세스의 순서, 화면 하단에는 간트 차트(Gantt Chart)를 준비하여 진행되는 상황을 시각적으로 표시하였다. 이 프로그램의 특징 중 하나는 선점형과 비선점형으로 나누어 구현하였는데, 이 기능은 SJFS, PS에만 적용되는데 현재 진행 중인 프로세스에 새로운 프로세스가 추가되는 경우 이 프로세스를 바로 스케줄러에 반영해서 실행(선점형)하거나, 다음 프로세스와 비교해서 실행(비선점형)하는 방법을 구분하여 실행하도록 그림 11과

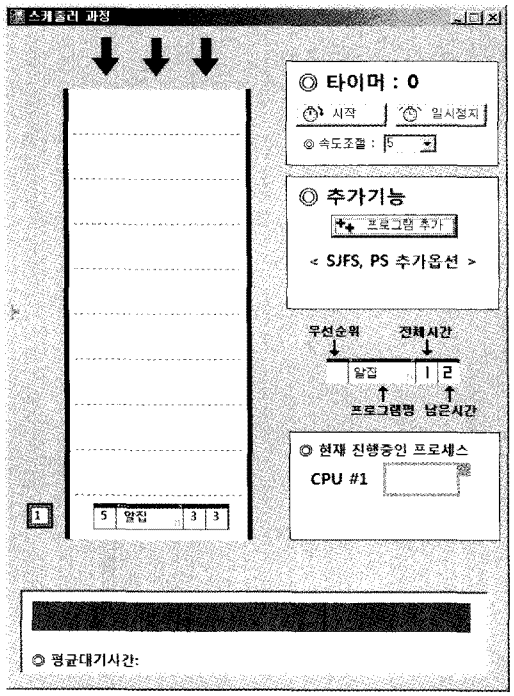

그림 10. 스케줄러의 주 실행 화면

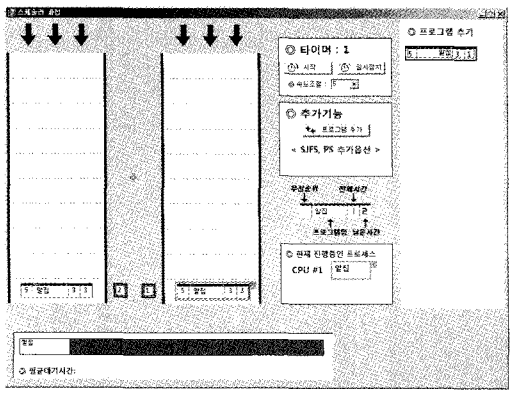

그림 11. 프로세스의 추가 화면

같이 구현하였다.

이러한 알고리즘 구현으로 선점형과 비선점형의 구현이 가능해졌으며, 프로그램이 종료된 후에는 프로세스의 수행한 결과를 다른 스케줄러의 방법과 수행 시간을 비교해 볼 수 있도록 하였으며, 그래프 형식인 간트 차트를 이용하여 구현하였다.

두 번째로 스케줄러의 전체 과정을 생략하고 기본 알고리즘(네 가지)을 비교할 수 있다. 위의 그림 12의 윗부분에서 제시한 스케줄링 알고리즘들 간을 비교할 수 있도록 차트로 도식화하였으며, 다양한 비교

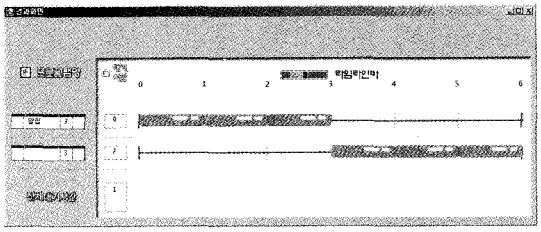

그림 12. 간트 차트의 구현 화면

분석이 가능하도록 위/아래에 각각 선택적으로 표시하여, 서로 다른 스케줄링 알고리즘들을 한 눈에 비교하도록 대조적인 인터페이스를 이용하였다. 또한 RR 알고리즘 경우 시간 쿼팀(time quantum)를 실시간으로 변경할 수 있게 하였다.

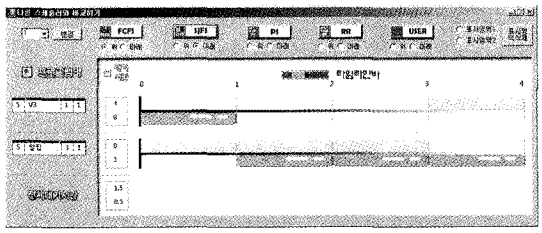

그림 13. 타 스케줄러와의 비교

이와 같이 구현된 결과물을 통해 학습자는 하나의 화면에서 여러 가지의 스케줄러를 비교/분석하고, 각각의 스케줄러의 특성과 장/단점에 대해 정확하고 빠르게 이해하며, 자기가 생각하고 있는 최적의 스케줄러와 우선순위의 스케줄러, 사용자 임의의 스케줄러에 대해서도 시뮬레이션 해 볼 수 있는 기회를 제공 받는다.

실제로 사용되는 알고리즘은 SQL 문과 포인터 형식으로 구현하여 처리하도록 설계하였다. SQL문은 현재 웹 상에서 널리 사용이 되고 있으며, Oracle, My-SQL, MS-SQL 등이 제공한다. 여기서는 파일 형식의 하나인 MS-Access 형식의 SQL문도 사용했으며, 이것은 응용 프로그램은 물론 웹 상에서도 쉽게 사용될 수 있다는 이점을 가지고 있기 때문이다.

기본적인 알고리즘은 SQL문을 통해 데이터베이스에서 미리 정의하고, 정렬된 형태의 순서를 가지고 스케줄러의 실행 순서를 결정하며(그림 14), 임의의 공간에 삽입하여 프로그램을 수행한다(그림 15).

이러한 실행 방식은 중간에 프로세스가 추가되는 경우에도 관리에 대한 편의성과 이동이 용이하도록

```
db_open();
String strsql = "SELECT O_SID, O_ID, [S_PROCESS], [S_ATTEN], [O_FCFS], [O_CHANGE], S_NAME, [S_MEMORY] FROM O
if (sel == "FCFS")
{
    strsql += " ORDER BY [O_CHANGE], [O_FCFS] ASC";
}
else if (sel == "SJFS")
{
    strsql += " ORDER BY [S_PROCESS], [O_FCFS] ASC";
}
else if (sel == "PS")
{
    strsql += " ORDER BY [S_ATTEN], [O_FCFS] ASC";
}
else if (sel == "USER")
{
    strsql += " ORDER BY [O_USER], [O_FCFS] ASC";
}
```

그림 14. 기본 스케줄러

```
}
// 자원: 과정 자원_결과값에 반영
public class vp_pds
{
    public String vp_id; //리스트 아이디
    public int vp_pro; //프로그램 전체 처리시간
    public int vp_mem; //프로그램 전체 처리자원
    public int vp_atten; //프로그램 우선순위
    public String vp_sid; //리스트 아이디;
    public int vp_etime; //완료된 시간
    public int vp_wtime; //대기했던 시간
    public int vp_start; //리스트에서 자기의 위치_처음
    public int vp_end; //리스트에서 자기의 위치_끝
    public int vp_enter; //들어온시간 시간(처음인지 중간인지 확인)
    public int vp_fcfs; //리스트에서 자기의 들어온 순서
    public String vp_pend = "0"; //프로그램 처리률
    public int x;
    public int y;
```

그림 15. 프로그램 공간 할당

설계하였다. SQL문을 이용한 결과 값을 통해 곧바로 스케줄러에 해당하는 적절한 처리를 한다. 또한 하나의 데이터처럼 스케줄러 리스트를 미리 구현해 두어 능숙한 동작이 아니어도 손쉬운 시뮬레이션이 가능하도록 하였다. 여기서 따로 리스트를 만들어 둔 이유는 일종의 순서를 설정하는 단계로 스케줄러의 처리 방법을 미리 구현하여 곧 바로 실행이 가능하도록 하는 의미도 있지만, 추후에 결과 값의 계산과 선점, 비선점형으로 나누어 처리하는 경우 리스트 순서의 수정이 용이하도록 하기 위해서 아래의 그림 16과 같이 스케줄러가 가진 자원을 클래스로 정의하여 구현하였다.

```
}
// 자원: 리스트 자원
public class sch_resource
{
    public String i_id = ""; //프로그램명
    public String i_con = "C"; //지속여부
    public int i_count = 0; //처리된 수
    public int i_fcfs = 1; //프로세스의 순서
    public int i_next = 0; //다음으로 처리되는 프로세스
    public String i_name = ""; //프로세스 이름
    public String i_process = "F"; //처리여부 확인
    public int i_num = 0; //자기 번호
    public int i_pro = 0; //자기전체 프로세스
    public string i_add = "N"; //처음 실행된건지 나중에 실행된건지 확인
}
```

그림 16. 스케줄러의 리스트

이와 같이 미리 할당된 공간을 가지고 스케줄러를 실행하며, 각각의 프로그램 끝에는 다음에 수행할 프로세스를 안내하도록 되어 있다. 이것은 중간에 프로세스의 순서가 변경될 경우를 대비하여 준비하였다. 프로세스의 순서가 변경되는 경우는 SJFS, PS 스케줄링 방법에서 발생할 수 있는데 프로세스가 실행 중일 때 다른 프로세스가 추가되는 경우, 그 프로세스가 우선권을 가져야 한다. 따라서 현재 진행 상황을 저장하고, 프로그램의 추가 이벤트가 발생하면 현재 위치부터 비교가 가능(선점형)하도록 구현하였고, 비선점형인 경우에는 현재 프로세스를 제외한 다음부터 비교가 가능하도록 다음의 그림 17과 같이 구현하였다.

```
{
    if (rb_1.Checked == true && v_pds[sch_list[iel].i_num].vp_atten < v_pds[sch_list[enter_process].i_num]
    {
        //현재값을 이동시킨후 자기값으로 저장한다.
        sch_list[sch_list.Count - 1].i_next = enter_process;
        enter_process = iel;
        //저장후에 현재 위아래 위치도 변동한다.
    }
    else
    {
        int tmp = sch_list[v_pds[sch_list[enter_process].i_num].vp_end].i_next;
        int tmp3 = v_pds[sch_list[enter_process].i_num].vp_end;

        if (tmp == 0)
        {
            sch_list[v_pds[sch_list[enter_process].i_num].vp_end].i_next = iel;
        }

        else
        {
            int i = 0;
            while (i != 5)
            {
                //찾으면 강제로 종료
                if (v_pds[sch_list[tmp].i_num].vp_atten > v_pds[sch_list[iel].i_num].vp_atten)
                {
                    sch_list[tmp3].i_next = iel; //기존의 값을 다음에 자기가 하도록 수정
                    sch_list[v_pds[sch_list[iel].i_num].vp_end].i_next = tmp;
                    i = 5;
                }
                else
                {
                    tmp3 = v_pds[sch_list[tmp].i_num].vp_end;
                    tmp = sch_list[v_pds[sch_list[tmp3].i_num].vp_end].i_next;

                    if (tmp == 0)
                    {
                        sch_list[tmp3].i_next = iel;
                        i = 5;
```

그림 17. 선점 및 비선점 방법의 구현

5. 결 론

본 논문은 CPU 스케줄링에 사용하는 알고리즘들에 대한 교육용 프로그램 제작에 관한 내용을 다루었다. 초중고 컴퓨터 교과 시간에 운영체제를 교수함에 있어 스케줄러의 개념을 시뮬레이터를 사용하여 이해하기 쉽게 교습할 목적으로 시뮬레이션을 구현하였다. 구체적으로는 CPU 스케줄링에 관한 효과적인 교육을 위해 시뮬레이션 기법을 도입하였으며, 이를 위해 시뮬레이션 프로그램을 설계하고 구현하였다. 컴퓨터를 교육하기 위해 컴퓨터 운영체제 교과에 대한 학습은 필수적으로 교육되어야 한다. 현재의 운영체제 수업은 이론 수업으로 이루어져 있고 학습자가 잘 이해하지 못하거나 어렵게 느끼는 교과목이다. 본 논문에서는 학생들에게 교육용으로 사용 가능한 프로그램을 개발하여 운영체제 교과를 효과적으로 이해하도록 하였다. 특히 운영체제의 주요 이론 부분인 CPU 스케줄링을 시뮬레이션 하여 컴퓨터 운영체제

의 동작 원리를 잘 이해할 수 있는 프로그램을 개발하였다. 여전히 구현하여야 할 부분이 많고, 세밀하지 못한 부분, 기능상 추가할 많은 부분이 향후에도 계속하여 연구되어야 하지만 차근차근 하나씩 구현해 간다면, 운영체제뿐만이 아니라 다른 공학 분야에서도 활용될 수 있을 것으로 생각된다.

중등교육 현장에서 시뮬레이션을 이용해서 운영체제의 일부분인 스케줄러를 교육할 필요성에의 의해 그 효용 가치가 있는 것으로 생각되며, 향후에는 인터페이스 부분에서 더 정밀하면서도 한 눈에 들어오는 화면 배치를 구성하고, 프로그램에 풍선 도움말을 삽입하여 각 알고리즘에 대한 설명문도 추가할 예정이다. 알고리즘 부분에서는 기본적으로 사용되는 알고리즘 이외에도 다른 참고 논문을 참고해서 실제 구현해 실제 알고리즘을 파악하며, 데이터베이스의 웹 연동을 통해 어디서든 서버에 연결하면 자신의 정보를 확인하고 프로그램을 다운받아 구동되도록 연구를 진행할 계획이다.

참 고 문 헌

[1] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne, 3rd Edition, "Operating System Concepts," 홍릉과학출판사, 2004.

[2] 박성익, 임철일, 이재경, 최정임, 교육 방법의 교육공학적 이해, 교육과학사, 2007.

[3] 김재춘, 부재율, 소경희, 채선희, 교육과정과 교육평가, 교육과학사, 2007.

[4] 임규혁, 임웅, 교육심리학, 학지사, 2008.

[5] 김병찬, 분산 환경을 위한 실시간 프로세스 관리에 관한 연구, 수원대학교 대학원 전자계산학과 석사학위 논문, 1993.

[6] 이형철, 실시간 운영체제에서 EDF 스케줄링 알고리즘 및 스택 자원 정책의 구현, 강원대학교 대학원 컴퓨터정보통신공학과 석사학위 논문, 2001.

[7] 한상직, 시뮬레이션을 이용한 지식 기반형 스케줄러의 개발, 인하대학교 대학원 산업공학과 석사학위 논문, 1998.

[8] 김명렬, 정영식, C# 언어 프로그래밍 바이블 C# Bible, 홍릉과학출판사, 2004.

[9] 최재규, Visual C# .NET 2008, 영진닷컴, 2004.

[10] 김상형, 닷넷 프로그래밍 정복, 가메출판사, 2008.

[11] 안찬식, 오상엽, "의미 분석과 형태소 분석을 이용한 핵심어 인식 시스템," 멀티미디어학회논문지, 제13권, 제11호, 1589-1593쪽, 2010년 11월.

[부록 A] SW의 사용자 설명서

A-1. 메인 화면

메인 화면은 프로그램을 실행 시키면 처음 나오는 화면이며, 각각 세 분야로 구분되어 있다.

① 스케줄러 : 운영체제의 스케줄러에 대해 학습할 수 있습니다. FCFS, SJFS, PS, RR 알고리즘이 있다.

② 데이터베이스 : OSM 프로그램을 실행하기 전에 데이터베이스에 프로그램을 추가 / 삭제가 가능하다.

③ CPU 스케줄러 프로그램 실행 : 데이터베이스의 프로그램과 스케줄러를 가지고 실제 프로그램을 구동한다.

A-2. 스케줄러(Scheduler)

CPU 스케줄러에서는 스케줄러의 정의와 스케줄러의 종류에 대해 나타난다. 빨간 박스 안에 네 개의 버튼 중 하나를 클릭하면 스케줄러의 설명이 하단에 나타나게 된다.

A-3. 데이터베이스(Database)

데이터베이스에서는 OSM에 사용할 프로그램을 추가/ 삭제하는 기능을 가지고 있다. 총 세 가지로 구성이 되어 있으며, ①과 ②번은 프로그램 추가/삭제 관련 ③번은 결과 기록 화면이다. ① 프로그램 추가 : OSM 사용할 프로그램을 설정한다. 프로그램명, 요구 시간, 요구 자원, 우선순위를 입력한 후 "등록" 버튼을 클릭한다. ② 프로그램 : 현재 등록되어 있는 프로그램을 나타낸다. 선택한 후, DEL 키를 누르면 삭제된다. ③ 결과 확인 : 스케줄러가 진행된 후, 결과 값을 확인하는 화면이다.

A-4. CPU Scheduler(스케줄러)

데이터베이스에 있는 프로그램을 가지고 실제 스케줄러를 작동하기 위해서 프로그램 설정을 하는 메

뉴이다. 순서적으로 ① → ② → ③ 순으로 진행이 되며, 전 단계가 실행되기 전에는 선택할 수 없다. ① 하드웨어 설정 : ①번에서는 CPU 코어 수와 자원을 설정할 수 있다. 숫자를 선택한 후에 "변경" 버튼을 클릭하면 적용되며 반드시 "고정" 버튼을 클릭해야 한다. ② 스케줄러 설정 : CPU 스케줄링을 선택한다. 총 네 가지 방법이 있으며, RR 라운드 스케줄링의 경우, 스케줄러 종류를 선택한 후에 "RR 스케줄러 사용"에 선택하면 오른쪽 숫자에 맞게 프로세스가 시간을 할당 받게 된다. 선택한 후에는 반드시 "적용" 버튼을 클릭한다. ③ 프로그램 : 스케줄러를 적용할 프로그램을 선택한다. 기본적으로 우선순위, 프로그램명, 수행할 프로세스, 필요한 자원 등으로 구성되어 있으며, 프로그램명을 클릭하면 하단의 데이터베이스가 추가된다. ④ 스케줄러 테이블 : 선택된 프로그램을 보여주면서 사용자가 스케줄러를 직접 반영하는 역할을 한다. "스케줄러 시작" 전까지는 변경이 가능하다.

1) 대기 자료 전체 삭제 : 현재 스케줄러를 수행하기 위한 프로그램을 삭제합니다(실제 프로그램은 삭제되지 않습니다).
2) 순서 지정 : "스케줄러 설정"에서 "USER"를 선택한 경우 나타나며, 현재 등록되어 있는 프로그램 순서를 사용자 임의로 변경한다.
3) 스케줄러 시작 : 현재 스케줄러 테이블에 있는 자료를 가지고 동적으로 진행 화면을 보여준다.
4) 알고리즘 비교 : 따로 동적 화면 없이 각각의 알고리즘 결과 값만을 나타낸다.

A-4.1 스케줄러 진행 상황

선택한 프로그램과 스케줄러를 가지고 실제 CPU 실행되는 과정이다. 1) 프로그램 순서와 공간 : 프로그램의 순서(왼쪽 보라색)를 왼쪽 1, 2를 통해 나타내고 있으며, 밑에서부터 프로그램이 차곡차곡 쌓이게 된다. 2) 타이머 : "시작"과 "일시정지" 버튼은 CPU가 수행 중 일 때 언제든지 일시정지를 통해 멈출 수 있으며, 일시정지 때 프로그램의 추가가 가능하다. 3) 추가 기능 : SJFS, PS 스케줄러의 경우 중간에 프로세스가 추가되는 경우가 있는데, 이 기능은 일시정지 상태에서만 가능하다. 4) 현재 진행 중인 프로세스 : 현재 CPU가 수행하고 있는 프로세스를 나타낸다. 5) 간트 차트 : 실행이 완료된 프로세스가 차곡차곡 쌓인다.

A-4.2 프로세스 추가하기

선택한 스케줄러 이외에 진행 중에 프로세스를 추가할 경우 타이머의 "일시정지" 후에 이러한 기능을 사용할 수 있다. ① "일시정지" 타이머를 클릭하면 맨 오른쪽 화면에 "프로그램 추가" 화면이 나타난다. ② 추가하려고 하는 프로세스 명을 클릭하면 맨 왼쪽 녹색 상자 안에 차례대로 추가된다. ③ 이와 같이 추가하면, 선점형 혹은 비선점형인 경우를 고려해서 전체 프로세스의 순위에 변동이 생기게 된다. ④ 완료가 되었으면 타이머의 "시작" 버튼을 눌러서 실행한다.

선점형의 경우, 현재의 프로세스의 남은 수를 비교해서 스케줄러 순서가 결정이 되며, 비선점형인 경우, 현재 프로세스를 종료한 후에 다음 프로세스를 비교한다. 이 기능은 SJFS, PS에서 사용 가능하다.

A-4.3 스케줄러 결과 보기

스케줄러의 진행이 끝난 후의 결과를 볼 수 있다. 결과 화면은 중간에 추가된 경우에는 아이콘이 활성화되지 않는다. 결과 화면은 크게 두 가지로 구분할 수 있다. 1) 대기 시간 : 프로세스의 대기 시간을 나타내며, 2) 타임 라인바 : 프로세스가 언제 진행되었는지 나타낸다.

A-4.4 알고리즘 비교

CPU Scheduler(스케줄러)에서 "알고리즘 비교"(빨간 박스) 버튼을 누르면 나오는 화면으로 여기서는 현재 데이터베이스에 저장되어 있는 프로그램을 가지고 모든 경우의 알고리즘을 계산하여 한 눈에 보여 준다. FCFS의 위 아래 버튼을 선택하여 클릭하면 타임 라인바의 가운데선을 기준으로 위 / 아래에 FCFS의 결과 값이 나타난다.

다른 스케줄러와의 비교도 가능하며 왼쪽 상단에 있는 숫자는 RR 알고리즘의 시간 분배를 현재 상황에서 바로 변경이 가능하도록 구현하였다. 이밖에도 대기 시간을 스케줄러마다 확인이 가능하다.

정 성 균

2001년 3월~2007년 2월 전주대학교 컴퓨터공학과 공학사
2006년 9월~2010년 8월 (주)싸이버테크
2007년 9월~2010년 2월 전주대학교 교육대학원 컴퓨터교육 전공 공학석사
2010년 9월~현재 전라북도교통문화연수원 교육 담당
관심분야 : 운영체제, ICT, 자연언어처리시스템의 구현

이 상 곤

1994년 전주대학교 영어영문학과 학사
1996년 전북대학교 컴퓨터과학과 학사
1998년 전북대학교 전산통계학과 석사
2001년 일본 국립 도쿠시마대학교 지능정보공학과 박사
2002년~현재 전주대학교 컴퓨터공학과 부교수
관심분야 : 한국어 정보처리, 자연언어처리, 한글공학, 정보검색, 문서분류, 영상처리시스템의 구현