

운영체제

Assignment #1: MCKU

201910162 컴퓨터공학부 문무현

**Basic Design**

**1. PCB 및 기타 디자인**

Mcku.h 에서 라이브러리는 <stdio.h>와 <stdlib.h>를 사용합니다.

페이지와 프레임의 수, 최대 프로세스 개수, 그리고 프로세스의 상태를 나타내기 위한 값은 매크로를 통해 정의했습니다. Main 함수의 인자로 전달받은 프로세스의 개수가 최대 프로세스의 개수인 256개를 초과할 경우 256개의 프로세스만 생성하도록 하였습니다.

PCB 구조체에 추가한 멤버 변수는 struct pcb\* next와 int state, char phys\_alloc[]가 있습니다. Next 변수는 PCB를 연결 리스트로 구현하기 위해서 추가하였으며, state 변수는 프로세스의 상태(RUN or READY)를 나타냅니다. Phys\_alloc는 현재 프로세스가 물리 메모리에 할당된 인덱스를 나타냅니다.

PCB는 원형 연결 리스트로 구현하였습니다. 프로세스 파일을 읽고, PCB 객체를 초기화하고 연결 리스트에 삽입하는 과정은 ku\_proc\_init 함수에 구현되어 있습니다.

**2. Ku\_proc\_init 디자인**

파일을 읽는 과정은 두 단계로 이루어집니다. 프로세스 파일명을 가지는 proc.txt를 읽는 것이 첫 번째이며, 프로세스를 의미하고 va 값을 가지는 proc\_n.txt 파일들을 n번 읽는 것이 두 번째입니다. n개의 프로세스 파일을 읽으며 PCB 객체도 n개 생성합니다. Pid는 0번부터 차례대로 부여하며, fd 값은 파일 디스크립터가 들어갑니다. PCB의 pgtable은 16 \* 1byte 크기로 동적할당 합니다. Phys\_alloc[] 배열은 0으로 초기화합니다. Next 변수를 통해 원형 연결 리스트 형태가 되도록 만듭니다. 이를 위해 struct pcb의 포인터인 head와 tail 전역 변수를 사용하였으며, 새로 생성되는 PCB 노드는 꼬리에 삽입되도록 하였습니다. 따라서 pid는 head로부터 오름차순으로 정렬됩니다. 원형 연결리스트이기 때문에 꼬리는 항상 머리로 돌아오며 next가 NULL 값이 되는 경우는 없습니다.

PCB 구조체의 초기화와 연결 리스트 구현이 끝나면 extern으로 선언된 current와 ptbr 변수를 첫번째 프로세스의 값으로 초기화합니다. 또한 첫번째 프로세스의 state를 RUNNING으로 만듭니다.

**3. ku\_pgfault\_handler 디자인**

페이지 폴트가 발생 시 처리는 ku\_pgfault\_jandler에 구현되어 있으며, 주소 변환 과정은mcku.c에 구현되어 있는 ku\_traverse 함수와 같지만, 변환된 주소의 valid bit(최하위 비트)를 1로 세팅하여 매핑되었음을 표시하였습니다.

또한 물리 메모리 공간을 나타내는 free\_list[]의 빈 공간을 찾아 할당하도록 하였습니다. Free list[]는 단순한 배열 형태이며, 페이지 프레임이 할당된 공간은 1, 비어 있는 공간은 0으로 표시합니다. 이 때 현재 프로세스의 페이지 테이블 엔트리가 물리 메모리 상에 어느 프레임을 차지하는지를 phys\_alloc[] 배열을 통해 저장합니다. Phys\_alloc[] 배열의 인덱스는 물리 메모리(free\_list[] 배열)의 i번째 프레임에 할당되었음을 의미합니다.

**4. ku\_scheduler 디자인**

스케줄러에서 다음 실행될 프로세스를 찾는 것은 PCB 연결 리스트를 통해 이루어집니다. 연결 리스트의 머리가 NULL 값인 경우 리스트가 비어 있는 상태이므로(=모든 프로세스가 종료된 상태이므로) 프로그램을 종료하기 위해 current에 NULL 값을 할당하고 ku\_scheduler 함수를 반환합니다. Current가 NULL이 되었으므로 mcku.c 파일의 ku\_run\_cpu 루프에서 프로그램이 종료됩니다.

실행될 프로세스의 탐색하는 과정에서 함수의 인자로 전달받은 pid 값은 사용하지 않습니다. While 루프 안에서 연결 리스트를 따라 가면서 PCB의 state 변수가 RUNNING인 노드를 찾습니다. RUNNING 상태인 노드를 찾으면 while 루프를 빠져나와 현재 노드의 state를 READY로 바꾸고, 다음 번 노드의 state를 RUNNING으로 바꿉니다. 새롭게 RUNNING 상태가 된 PCB로 current와 ptbr 값을 변경합니다.

While 루프 안에서 RUNNING 상태인 PCB를 찾지 못하는 경우가 있습니다. 실행 중인 프로세스가 종료되어 RUNNING 상태의 프로세스는 없고, 모두 READY 상태인 프로세스만 존재하는 상태에서 스케줄러가 호출된 경우입니다.

이 때 PCB는 원형 연결 리스트이기 때문에 포인터가 머리로 돌아오면 모든 노드에 RUNNING 상태가 없다는 것을 알 수 있습니다. 이 때 종료된 프로세스의 다음 실행 순서를 찾기 위해 추가 변수들을 사용합니다. 다음 실행될 프로세스는 (last\_terminated + count) % NUM\_PROCESS로 구합니다. NUM\_PROCESS는 main 함수의 인자로 전달된 프로세스의 개수이고, Last\_terminated 변수는 가장 최근 종료된 프로세스의 pid를 가집니다. count 변수는 리스트를 한 번 순회할 때마다 1씩 증가하는 변수입니다. (last\_terminated + count) % NUM\_PROCESS 값이 현재 노드의 pid와 일치하면 현재 노드의 프로세스가 다음 실행할 프로세스가 됩니다. Count의 값이 증가하는 이유는 현재 프로세스와 다음 프로세스 사이에 종료된 프로세스가 있는 경우를 고려하였기 때문입니다. 0, 1, 2, 3 프로세스 중 2번 프로세스가 종료된 상태에서, 1번 프로세스가 종료되었다면 다음 번 실행 순서인 3번 프로세스는 최근 종료된 프로세스의 pid에 1만 더하여서는 찾을 수 없기 때문입니다.

**5. ku\_proc\_exit 디자인**

프로세스 종료 시 연결 리스트를 순회하여 종료된 프로세스의 pid값과 일치하는 노드를 찾습니다. 현재 노드를 가리키는 cur\_pcb와 이전 노드를 가리키는 prev\_pcb를 사용합니다. 두 변수는 지역 변수이며 head와 tail 변수는 삽입 때 사용한 전역 변수입니다. 종료할 프로세스의 pid와 같은 노드를 찾을 때까지 cur\_pcb와 prev\_pcb를 변경하며 리스트를 순회합니다. 삭제할 노드를 cur\_pcb로 만듭니다. 이 때 prev\_pcb가 초기 값인 NULL 값에서 변하지 않은 경우 삭제할 노드는 머리 노드가 됩니다. 원형 연결 리스트이기 때문에 머리 노드가 바뀐 경우 꼬리 노드가 머리를 가리키도록 포인터를 다시 지정해야 합니다. Cur\_pcb의 next가 head인 경우 꼬리 노드를 삭제하는 것입니다. 따라서 tail 포인터를 다시 지정해야 합니다. 중간 노드인 경우 prev\_pcb의 next를 cur\_pcb의 next로 연결함으로써 cur\_pcb를 삭제합니다.

노드를 연결 리스트에서 제거했다면 이후 물리 메모리를 의미하는 free\_list[]를 업데이트하고, 동적할당 되었던 메모리 자원을 회수하는 과정이 필요합니다. 먼저 free\_list[]에서 종료된 프로세스가 차지하고 있던 프레임들을 phys\_alloc[] 배열을 통해 해제하는 것부터 시작합니다. 이후 last\_terminated 변수를 업데이트하여 RUNNING 상태의 프로세스가 종료된 경우이더라도 스케줄러에서 다음 실행될 프로세스를 찾을 수 있도록 합니다. PCB의 pgtable은 동적할당 되었으므로 free합니다. 또한 PCB의 fd를 이용하여 열려 있던 파일을 닫습니다. 마지막으로 PCB 객체를 삭제합니다.

프로세스가 종료될 때마다 TERMINATED 전역 변수가 0부터 1씩 증가하도록 하였습니다. 이는 NUM\_PROCESS 값과 비교하여 모든 프로세스가 종료되었는지 확인하기 위함입니다. 모든 프로세스가 종료된 경우 연결 리스트의 머리 포인터를 NULL 값으로 바꿉니다. 따라서 다음 스케줄러가 호출되었을 때 current가 NULL 값이 되며 ku\_run\_cpu 루프 안에서 프로그램이 종료될 수 있도록 만듭니다.

**Description for important functions**

주어진 네 개의 함수 이외에 추가적인 함수는 정의하지 않았습니다.

|  |  |  |
| --- | --- | --- |
| Ku\_proc\_init | Functionality | 파일 읽기 및 PCB 초기화 |
| Parameters | Int nprocs, char\* flist |
| Return Value | 없음 |

|  |  |  |
| --- | --- | --- |
| Ku\_pgfault\_handler | Functionality | 페이지 폴트 발생 시 처리(pte 할당) |
| Parameters | Char va |
| Return Value | 없음 |

|  |  |  |
| --- | --- | --- |
| Ku\_scheduler | Functionality | 라운드 로빈 방식으로 프로세스 스케줄링 |
| Parameters | Char pid |
| Return Value | 없음 |

|  |  |  |
| --- | --- | --- |
| Ku\_proc\_exit | Functionality | 물리 메모리와 PCB 업데이트 및 메모리 해제 |
| Parameters | Char pid |
| Return Value | 없음 |