운영체제 과제 #1. CFS-like scheduling policy, KU\_CFS

컴퓨터공학과 201511269 송상연

**Basic Design**

Node 구조체를 다음과 같이 선언한다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8 | typedef struct tagNode {      double vruntime;      double DeltaExec;      double weight;      int count;      pid\_t pid;      struct tagNode\* NextNode;  }Node; | [cs](http://colorscripter.com/info#e) |

링크드리스트의 한 노드를 하나의 프로세스와 관련된 속성들의 집합이라고 했을 때, 각 노드에 들어가는 속성은 다음과 같다.

double vruntime : 총 실행 시간(vruntime += DeltaExec \* weight)

double DeltaExec : 지난 vruntime 업데이트 후 흐른 시간

double weight :

int count : 실행한 time slice 개수(argv[6]로 받은 총 time slice 수만큼 실행했을 경우 SIGKILL신호를 보내기 위해 필요하다)

pid\_t pid : 해당 프로세스의 pid값

struct tagNode\* NextNode : 다음 노드의 주소 값

**main함수에서 하는 일**

인자로 받은 argv[1], argv[2], argv[3], argv[4], argv[5]만큼 프로세스를 생성한 후,

parent 프로세스에서는 vruntime = 0, DeltaExec = 0, 인자 위치에 맞게 nice\_value를 -2~ 2까지 부여하여 계산된 weight값, count = 0, child 프로세스의 pid를 인자값으로 넘겨 노드를 생성한다. 그리고 그 노드들을 연결한다.

child 프로세스에서는 ku\_app “알파벳”으로 execl을 실행한다. 알파벳은 생성된 프로세스 수만큼, 생성된 순서대로 A부터 들어간다.

sleep(5)로 ku\_app.c의 소스 중 kill(getpid(), SIGSTOP)부분이 제대로 완료되게끔 5초 기다린다.

setitimer로 설정한 시간(여기서 하나의 time slice실행 시간은 1초)만큼 timer\_handler함수가 주기적으로 실행된다. timer\_handler함수에서는 vruntime값이 적은 순서대로 링크드리스트를 정렬한 후, 링크드리스트의 head노드(제일 vruntime값이 적은 노드)의 pid에게 SIGCONT신호를 보낸다. 그 후, 해당 노드의 DeltaExec에 time slice 실행 시간(여기서 1초)을 대입하고, 총 실행 시간을 대입한 DeltaExec에 따라 식에 맞게 업데이트 한다. 그리고 해당 노드의 count(실행한 time slice 개수)를 1증가시킨다. 이제 총 실행 시간도 업데이트 했으니, 다시 DeltaExec을 0으로 초기화한다. 그리고 그 프로세스에게 SIGSTOP신호를 보낸다. 이 과정이 1초 주기로 반복된다.

만약 해당 노드의 count수가 argv[6]로 받은 총 time slice 개수랑 같아지면, 해당 노드를 링크드리스트에서 삭제하고 해당 pid에게 SIGKILL신호를 보낸다. 이렇게 SIGKILL신호를 보낸 프로세스 수와 생성한 프로세스 수(인자로 받은 argv[1]~argv[5]의 합)가 같아지면 주기적으로 timer\_handler함수를 실행하던 것을 멈추고 종료한다.

2>output.txt에 출력된 알파벳들을 통해 실행된 프로세스들의 순서를 알 수 있다. 프로세스들에게 부여된 nice value에 따라 vruntime값이 정해져 nice value값이 큰 프로세스들은 nice value값이 작은 프로세스들과 실제 실행한 시간이 같아도 vruntime값이 크게 계산되어 시간이 지날수록 스케줄러의 우선순위에서 밀려나 늦게 실행되는 것을 볼 수 있다.

**Description for important functions**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8 | Node\* create\_node(double vruntime, double DeltaExec, double weight, int count, pid\_t pid);  void destroy\_node(Node\* Node);  void append\_node(Node\*\* Head, Node\* NewNode);  void remove\_node(Node\*\* Head, Node\* Remove);  Node\* get\_node\_at(Node\* Head, int Location);  int get\_node\_count(Node\* Head);  void sort\_linkedlist(Node\* Head);  void time\_handler(int signum);  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info" \l "e" \t "_blank) |

|  |  |
| --- | --- |
| Node\* create\_node(double vruntime, double DeltaExec, double weight, int count, pid\_t pid); | |
| Functionality | 새로운 노드를 Node구조체의 포인터 변수로 생성한다. 그리고 생성한 노드의 각각의 속성들에 매개변수로 받은 값을 넣어준다. 처음에 생성한 프로세스 수만큼 노드를 생성할 때 필요하다. |
| Parameters | 노드의 속성에 들어갈 값(double vruntime, double DeltaExec, double weight, int count, pid\_t pid) |
| Return Value | 새로 생성한 노드 |

|  |  |
| --- | --- |
| void destroy\_node(Node\* Node); | |
| Functionality | malloc으로 동적 메모리를 할당했던 노드들 중 인자로 받은 노드를 free로 해제한다. |
| Parameters | free할 노드(Node\* Node) |
| Return Value | 없음 |

|  |  |
| --- | --- |
| void append\_node(Node\*\* Head, Node\* NewNode); | |
| Functionality | create\_node()로 만든 새로운 노드를 링크드리스트에 연결한다. 생성한 프로세스 수만큼 노드를 생성한 후, 노드들끼리 연결할 때 필요하다. |
| Parameters | 링크드리스트의 Head노드와 새로 연결할 노드(Node\*\* Head, Node\* NewNode) |
| Return Value | 없음 |

|  |  |
| --- | --- |
| void remove\_node(Node\*\* Head, Node\* Remove); | |
| Functionality | 링크드리스트에서 Remove노드와의 연결을 해제한다. 종료된 프로세스에 해당하는 노드를 해제하기 위해 필요하다. |
| Parameters | 링크드리스트의 Head노드와 제거할 노드(Node\*\* Head, Node\* Remove) |
| Return Value | 없음 |

|  |  |
| --- | --- |
| Node\* get\_node\_at(Node\* Head, int Location); | |
| Functionality | Location(Head노드(0)로부터 몇번째)에 있는 노드를 가져온다. vruntime이 적은 순서대로 정렬했을 때, 첫번째 노드(Location이 0인 노드)를 가져오기 위해 필요하다. |
| Parameters | 링크드리스트의 Head노드와 그 Head로부터 몇번째인지 알려주는 Location변수(Node\* Head, int Location) |
| Return Value | 링크드리스트에서 Location변수 위치에 해당하는 노드 |

|  |  |
| --- | --- |
| int get\_node\_count(Node\* Head); | |
| Functionality | 링크드리스트의 전체 노드 수를 반환한다. 링크드리스트에서 전체 노드의 속성 값들을 보기 위해 for문으로 출력할 때 필요하다. |
| Parameters | 링크드리스트의 Head노드(Node\* Head) |
| Return Value | 전체 노드 수 |

|  |  |
| --- | --- |
| void sort\_linkedlist(Node\* Head) | |
| Functionality | Bubble Sort로 링크드리스트를 정렬한다. 일정 주기마다 vruntime이 적은 순서대로 링크드리스트를 정렬할 때 필요하다. |
| Parameters | 링크드리스트의 Head노드(Node\* Head) |
| Return Value | 없음 |

|  |  |
| --- | --- |
| void time\_handler(int signum); | |
| Functionality | setitimer에서 설정한 주기마다 실행하는 함수이다. 여기서 sort\_linkedlist()로 정렬해서 링크드리스트의 Head에 오는 노드에게 SIGCONT신호를 보내고 링크드리스트에서 SIGCONT신호를 보낸 프로세스의 노드의 vruntime값을 업데이트하고 SIGSTOP을 보내는 역할을 한다. |
| Parameters | 시그널 번호(int signum) |
| Return Value | 없음 |