# 운영체제

-linked list 구현-

소프트웨어학과

201520860

노근탁

**구현 과정** : 먼저 리스트에 필요한 노드 구조체와 전체적인 리스트의 구조체를 정의했습니다. 노드 구조체는 해당 노드가 다음 노드를 가리키는 연결리스트 구조이기 때문에 struct mylist_node * nextnode 로 다음 노드를 가리키는 포인터를 구현하고 노드에 들어가는 데이터 값은 정수를 저장하기때문에 int 로 선언했습니다. List 구조체는 head 노드를 가리키는 Head 포인터, 연산에 필요한 cur과 삭제연산에 필요한 cur이전의 노드를 가리키는 before 포인터를 정의했습니다.

**mylist_init** 함수는 초기화할 리스트의 주소를 받아 mylist 구조체에 정의된 포인터들을 NULL값으로 초기화하도록 구현했습니다.

**mylist_destroy** 함수는 모든 노드를 삭제할 리스트의 주소를 받아서 for 반복문으로 리스트 전체를 조회하며 기존에 정의된 mylist_remove함수로 노드를 제거하도록 구현하였습니다

**mylist_insert** 함수는 리스트의 주소와 새 노드가 삽입될 위치 바로 전의 노드, 데이터 값을 전달받아서 새노드를 malloc으로 메모리할당해 이후에 메모리 해제가 가능하게 하였습니다. 이전에 생성된 노드가 없으면 새로 생성되는 노드를 head로 지정하게 하였고 이전에 생성된 노드가 있으면 그 다음에 삽입될 수 있도록 하였습니다. 이 과정에서 리스트의 중간에 삽입될 경우 새 노드가 삽입되어도 리스트가 끊기지 않게 구현했습니다.

**mylist_remove** 함수는 리스트의 주소와 삭제할 타겟 노드 주소를 전달 받아 수행되며 target 이 NULL값일 경우 함수가 끝나도록 예외처리 하였고 target이 head 노드이면 head 다음 노드를 head로 지정 해준 뒤 free함수가 수행되도록하여 리스트의 구조가 유지되도록 하였습니다. 그 이외의 경우에는 mylist_find 함수로 before값을 지정해주고 before노드의 nextnode 값을 target노드의 다음 노드를 가리키도록 하여 리스트 구조가 유지되도록 하였습니다.

**mylist_find** 함수는 target값을 가지는 노드를 찾는 함수로 cur 포인터를 head로 초기화해서 head 부터 차례로 리스트를 조회해서 찾도록 while 반복문으로 구현했습니다. 이 과정에서 target 노드를 찾으면 그 노드를 가리키는 cur 포인터를 반환하고 찾지못하였으면 remove 함수 연산에서 사용되는 before 포인터도 값을 지정해주었습니다.

**mylist_get_head** 함수는 전달받은 리스트의 헤드값을 반환하고 헤드가 없을 경우 NULL값을 반환하도록 했습니다.

**mylist_print** 함수는 전달받은 리스트를 for 반복문을 이용해 head부터 차례대로 조회하여 각 노드들의 데이터를 출력하도록 하였습니다.

**고찰** : 이번 과제를 하기전에 가상머신을 통해 리눅스 환경을 설정하는 방법과 git을 이용해서 타인과 개발환경을 공유할 수 있는 방법을 배웠습니다. 그리고 코드를 컴파일하는 과정에서 필요한 최소한의 리눅스 명령어들에 대해 숙지하였습니다. 메인과제를 통해서는 기존에 학습했던 C언어와 자료구조에 대해 복습하고 리눅스와 C언어에대해 더 많은 공부의 필요성을 느끼는 계기가 되었습니다.

**피드백** : 리눅스 환경 구현 설명해주신 영상에서 별도의 설명이나 이런 것이 다소 부족해서 조금 아쉬웠습니다. 저처럼 리눅스 환경이나 git 사용이 거의 처음인 경우에는 어려움이 있을 것 같습니다. 이외에 groups를 사용해서 질문을 공유하는 과정은 매우 도움이 많이 되었습니다.