1️⃣ OS_실습

Vim Plugin

- Vim vundle : vim plugin 들을 관리, 온라인으로 목록 불러와서 설치, 설정까지 가능
  - .vimrc 생성
  - 위치 : home
  - .vimrc 필요한 plugin 및 option 설정
  - Nerdtree
    - Plugin ‘scrooloose/nerdtree’

      nmap <F9> : NERDTreeToggle <CR>
  - Jellybeans
    - colorscheme jellybeans

      syntax on
  - Rainbow
    - Plugin ‘frazrepo/vim-rainbow’

      let g:rainbow_active=1
  - Indent-guides
    - Plugin ‘nathanaelkane/vim-indent-guides’

      let g:indent_guides_enable_on_vim_startup=1
  - Vim-Airline : 현재 작업정보 보여줌
    - Plugin ‘vim-airline/vim-airline’
- Vim option
  - syntax on : 문법 강조
  - set nu : 줄 번호 표시
  - set cindent : 자동 들여쓰기
- ctags : 소스 파일 내에 정의된 전역 변수, 함수, 매크로, 구조체의 ㅈ어보를 데이터베이스 파일로 생성
  - yun install ctags : 설치
  - ctags -R : db 생성
  - 경로 설정(위치 vimrc) : set tags=위치경로/tags
- cscope : ctags로 찾기 힘든 전역 변수나 함수의 호출 등을 볼 때 사용
  - yum install cscope
  - shell script 생성

Shell Script

- shell script : Unix, Linux 등에서 사용하는 일반적인 명령어나 if나 for와 같은 프로그래밍적인 요소로 이루어진 interprinter 기반의 script언어
- 기초 문법
  - shell script 맨 윗줄에 #!/bin/bash 붙임
  - echo : 문자열을 terminal에 출력하는 명령어
  - chmod +x myshell.sh → vim파일인 myshell.sh에 실행 권한 줌

    - ./myshell.sh → 실행
- 변수 사용
    - 변수 선언 : $변수
    - 전역 변수(모두 다 알고 있는), 지역 변수
    - 예약 변수, 환경 변수 : 시스템에서 미리 정해둔 변수들이 존재
        - HOME : 사용자 home directory
        - PATH : 실행 파일 찾는 directory 경로
        - PWD : 현재 작업 중인 directory 경로
        - USER : 사용자 이름
        - OSTYPE : OS 종류
    - 매개변수 : ./vim.sh 매개변수1 매개변수2
    - ${변수 := 문자열} : 문자열을 변수로 치환
    - 문자열 패턴 변경
        - ${변수#패턴} : 변수 앞에서부터 처음 찾은 패턴과 일치하는 패턴 앞 모두 제거
        - ,, ##,, : 앞에서부터 마지막으로 찾은 패턴과 일치하면 패턴 앞 모두 제거
        - ,, %,, : 뒤에서 부터 처음 찾은 패턴 일치하면 패턴 뒤 모두 제거
        - ,,%%,, : 뒤에서 부터 마지막으로 찾은 패턴 일치하면 패턴 뒤 모두 제거
        - #변수: 변수의 길이
        - 변수/찾는문자열/바꿀문자열 : 변경, 없으면 삭제
        - 변수/#찾을문자열/바꿀문자열 : 문자열 시작과 같으면 문자열 변경(앞에서부터 1개만)
        - 변수/%,, : 문자열 뒤와 같으면 문자열 변경
    - 조건문
        - if 조건식 then echo~ elif 조건식 then echo~ else echo~ fi
        - case $변수 in 조건값) echo~~;; 조건값) echo~;; *) esac
    - 반복문
        - for 변수 in 범위 do echo~; done
        - while [$변수 연산자 $변수] do echo~ done
- grep : 특정 디렉토리, 로그 등에서 특정 문자열 찾는 명령어 → grep -i cscope list.txt
- find : 파일을 찾아주는 명령어 → find ./ -name list.txt
- awk : 특정 index 문자열 출력
- user 계정 만들기

```
cat /etc/passwd | grep jeong_min | wc-l
```

Lock

- Lock programming : Lock, Unlock ,, Multi process(thread)
- Multi thread
    - vim thread.c
    - gcc thread.c -lpthread -o thread.out (compile)
- Multi thread + Lock
    - cp thread.c thread_lock.c

    - pthread_mutex_t lock; →추가
    - main함수안에 pthread_mutex_init(&lock, NULL); → 초기 lock
    - static void 안에 pthread_mutex_lock(&lock); pthread_mutex_unlock(&lock);
- binary semaphore
    - vim thread_bin_sen.c
    - pthread_mutex_t lock;

      sem_t semaphore;
    - main 함수안에

      ```
      pthread_mutex_init(&lock, NULL);
      sem_init(&semaphore, 0, 1);
      ```
    - void함수안에

      ```
      pthread_mutex_lock(&lock)
      sem_wait(&semiphore);

      pthread_mutex_unlock(&lock);
      sem_post(&semaphore)
      ```
- counting semaphore

  ```
  #define SEM_COUNT 3
  int working[SEM_COUNT];

  #main함수
  pthread_mutex_init(&lock,NULL);

  sem_destroy(&semaphore);

  #static void
  pthread_mutex_lock(&lock);
  for(int i=0; i<SEM_COUNT; i++){
      if(working[i]==0){
      working[i]=1;
      count_index=i;
      break;
  }
  }
  pthread_mutex_unlock(&lock);
  ```
    - working에 대한 lock에 대해 각각의 thread는 각각의 count에 접근
- dead lock

  ```
  int first_count=0;
  int second_count=0;

  pthread_mutex_t first_lock;
  pthread_mutex_t second_lock;

  #main
  pthread_mutex_init(&first_lock,NULL);
  pthread_mutex_init(&second_lock,NULL);
  ```

```
#void_1
pthread_mutex_lock(&first_lock);
pthread_mutex_lock(&second_lock);

first_count++;
second_lock++;

pthread_mutex_unlock(&second_lock);
pthread_mutex_unlock(&first_lock);

#void_2
pthread_mutex_lock(&second_lock);
pthread_mutex_lock(&first_lock);

first_count++;
second_lock++;

pthread_mutex_unlock(&first_lock);
pthread_mutex_unlock(&second_lock);
```

Lock_2

- Lock : Data Structure
  - 자료 구조 : 정보를 효율적으로 저장하기 위하여
  - 데이터는 현재 볼 수 있는 정보이고 information은 우리가 원하는 것으로 가공한 정보이다.
  - 어떤 자료구조를 고르는지가 중요
  - 종류 → 자료구조 < 단순구조, 선형구조, 비선형구조, 파일구조 < 선형구조(리스트, 연결 리스트, 스택, 큐) < 비선형구조(트리)
- Time Complexity (시간 복잡도)
  - Queue는 모든 데이터를 원하는 것을 찾을 때까지 검색
    - Search=O(n) (모든 데이터), Insertion=O(1), Deletion=O(1)
  - Hash Table은 1개의 연산으로 모든 작업 끝남
    - Search=O(1), Insertion=O(1), Deletion=O(1)
  - Binary Search Tree : pointer나 algorithm을 통해 data 찾음 = B-Tree
    - O(logN)
- Time Complexity VS Space Complexity
  - Time Complexity는 Run time으로 비교, Space Complexity는 Memory Consumption
  - 최적의 좋은 자료 구조 찾기
- Hash
  - 데이터 관리, 저장, 검색 빠름, O(1)은 time complexity
  - 기본 동작 : Key K → Hash Function → Hash Address → Entry
  - Input 데이터 길이 상관없음 → output 데이터 고정길이
  - Hash function
    - input : data of any length, output : fixed length
    - 강력성을 위해 사용됨
    - h(x) = x mod m : x와 m을 나눈 나머지 (x:input, h(x):output, mod=%operation)
  - Hash Problem : 두 키 A, B가 있으면 h(A)==h(B) , 이 둘은 같은 위치여서 hash collusion이 발생함 (data이미 존재하여 data가 저장되지 않음)

    - Hash Solution : Chaining → separate chaining을 통해 정해진 위치에 data 쌓임
        - chaining을 분리
        - 충돌 위치에 entry 추가
        - linked list 사용
- Queue
    - Quere Data Structure
    - FIFO : 가장 먼저 들어올 데이터가 가장 먼저 나가는 방식
    - LIFO : 가장 마지막에 들어온 데이터 먼저 삭제
    - 뒤쪽으로 add(enqueue)
    - 앞쪽부터 delete(dequeue)
    - O(n) : search를 위한 시간 (빠르다)
    - O(1) : add and delete
    - queue가 꽉 차면 삭제하지 않으면 사용할 수 없다
    - 다른 type) circle queue, priority queue
    - lock을 거는 임계영역 설정범위 : enqueue하는 thread끼리, dequeue하는 thread끼리
- hash-queue problem
    - hash queue lock problem (insert) : hash function→find bucket loc→check entry→insert entry→insert node
    - ,,(delete) : hash function → find bucket loc → check entry → find target key → delete node → delete entry

Cloud Computing

- 클라우드 컴퓨팅 이점
    - 비용 절감
    - 민첩성, 탄력성, 전세계 배포
- AWS
    - amazon.com을 통해 얻은 경험으로 2006년에 서비스 시작
    - 클라우드 컴퓨팅을 통한 웹서비스 기반 제공
- what is cloud computing
    - IT 자원을 직접 설치하지 않고, 원격으로 필요한 만큼 빌려쓰는 컴퓨터 패러다임
    - 컴퓨팅 모델 - 메인 프레임 기반 컴퓨팅, 서버/클라이언트 컴퓨팅, 클라우드 컴퓨팅
- 필요성
    - 투자 및 비용 감소, 전력 비용 감소, 확장성 증가, 가용성 및 신뢰성 증가, 이동성 증가, 보안 강화
- examples
    - 클라우드 컴퓨팅 기반 기술 : 가상화 기술, 웹 기술, Amazon EC2, Google Cloud, Openstack