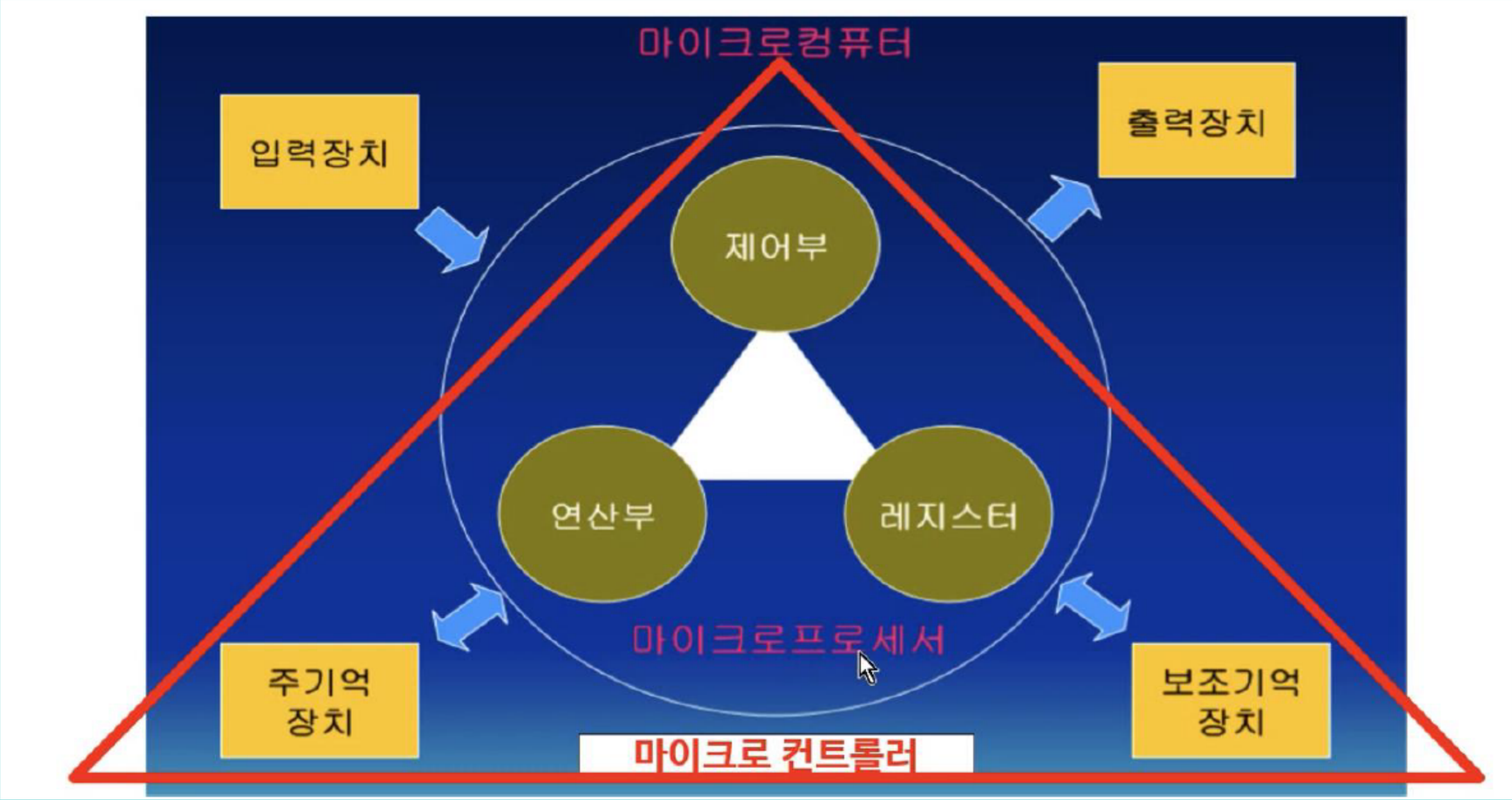
2023.03.15.(수)

마이크로프로세서

2402김나영

마이크로 컴퓨터/주기억 장치/보조기억 장치

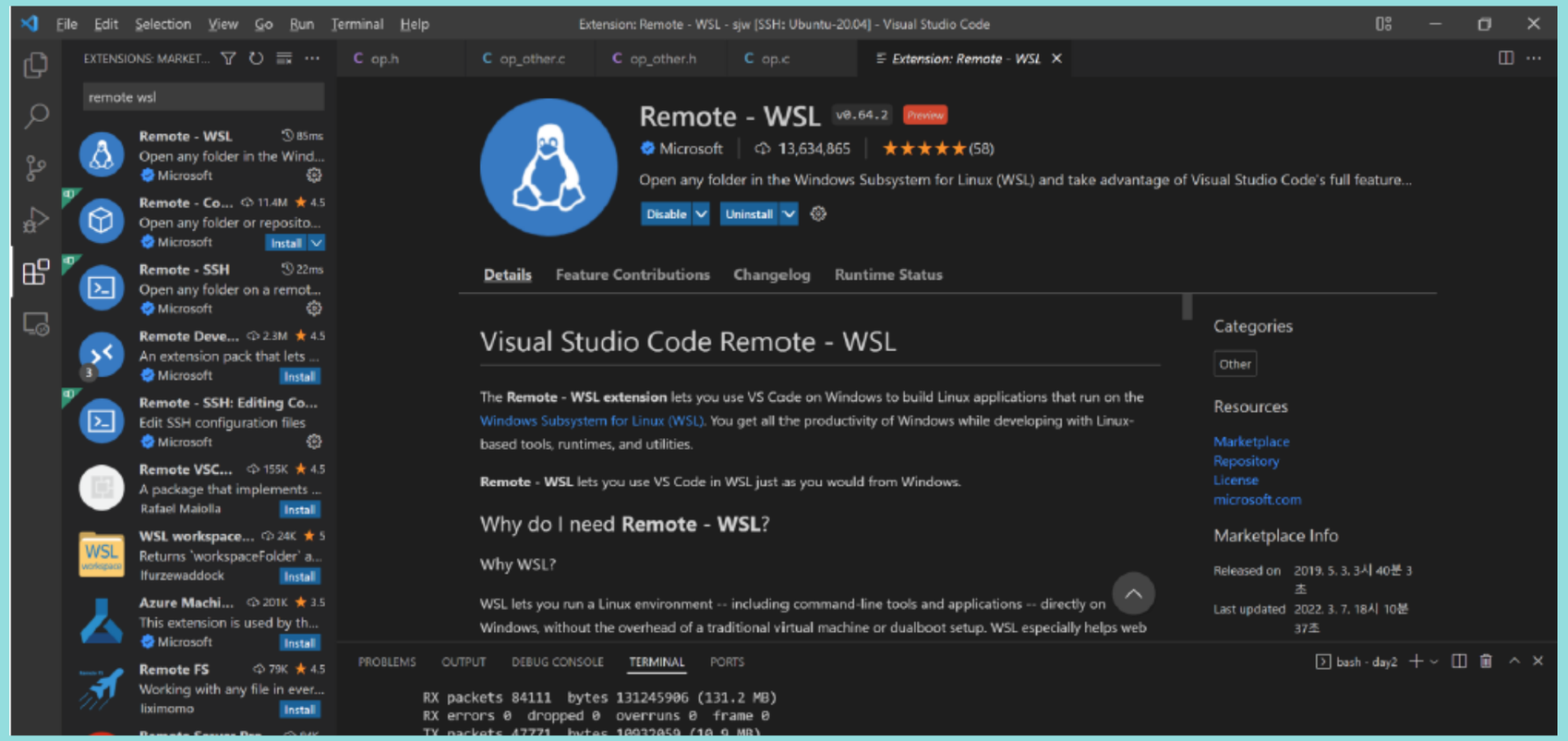


리눅스의 특징

- 이식성과 확장성이 용이
- 텍스트 모드 중심의 관리와 다양한 관리 환경 제공
- 풍부한 소프트웨어 개발 호나경 제공
- 다양한 네트워크 서비스 및 작업환경 지원
- 뛰어난 안전성
- 폭넓은 하드웨어 장치 지원
- 시스템의 높은 신뢰성
- 가격 대비 탁월한 성능(무료)

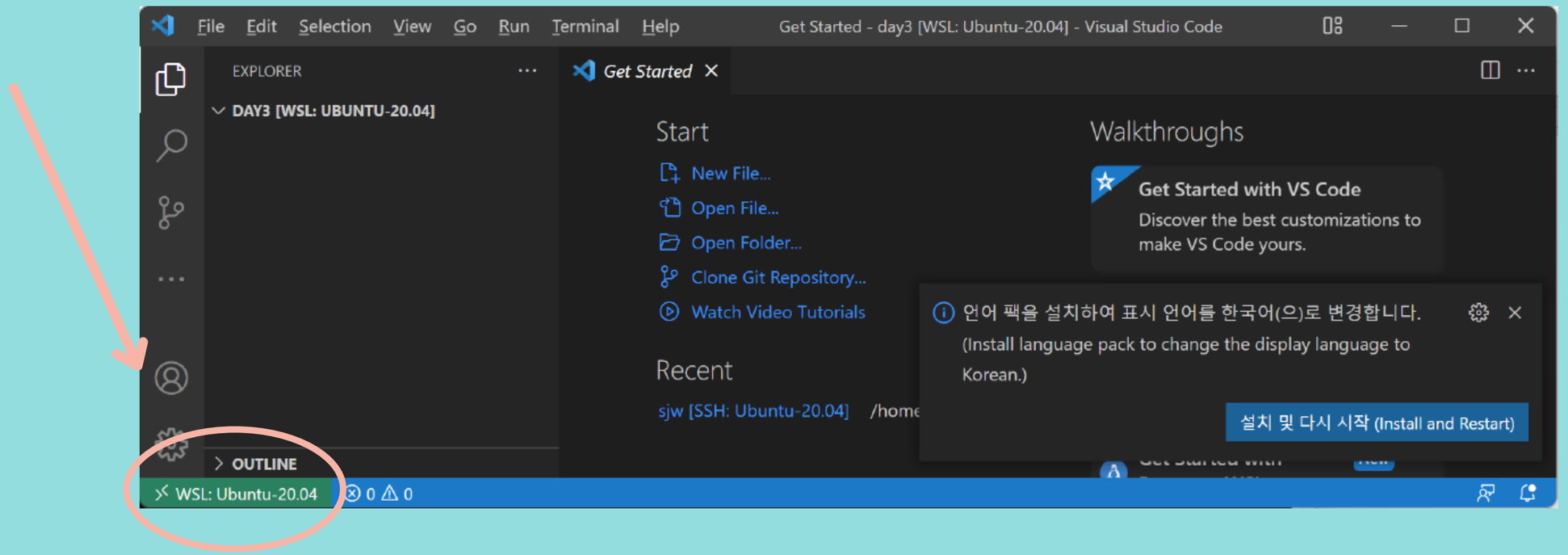
VSCode 에서 Remote-WSL 설치

- extension tab에서 Remote-WSL 추가 설치

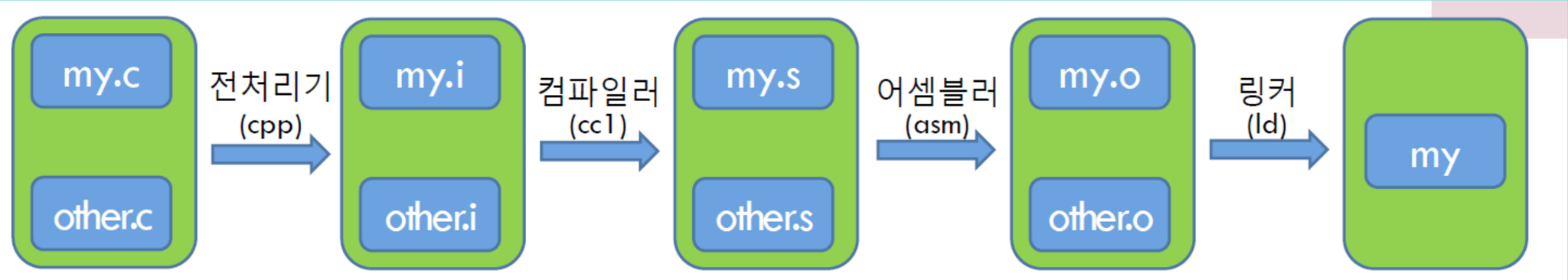


Ubuntu 에서 VSCode 실행

- Ubuntu에서 VSCode 실행
  $code . //VSCode 실행
- Ubuntu상에서 vscode 실행한 상태



C프로그램 생성과정



- 전처리기: 소스 파일에 gss를 동작시키면 가장 먼저 전 처리기 cpp가 동작하고, cpp는 소스 파일의 #include와 #define 과 같은 전처리기 부분을 처리
- 컴파일러: 전처리된 파일로부터 어셈블리어로 된 파일을 생성
- 어셈블러: 어셈블리어로 된 파일을 기계가 직접 이해할 수 있는 기계어로 된 오브젝트파일
- 링커: 링커목적 파일을 관련된 라이브러리와 연결하여 실행 파일 생성

gcc 실행방법

- 컴파일 방법
  $ gcc 소스파일 이름
  (a.out 파일 생성, a.out을 실행하기 위해선 $ ./a.out)
- 컴파일 할때 출력 파일 이름 지정 방법
  $ gcc -o 출력파일 이름 소스파일 이름
  (ex.$ gcc -o file file.c)
- 여러 파일을 동시에 컴파일 하는 방법
  $ gcc -o 출력파일이름 소스파일이름1, 소스파일이름2
  (ex. $gcc -o file file1.c file2.c)

포인터(1) - 포인터 변수 선언

mic_0315 > test1.c > main()

```c
#include <stdio.h>
int main(){
    int *ptr;                   //포인터 변수 선언
    int num = 10;               //int형 변수를 선언하고 10저장
    ptr = &num;                 //ptr에 num의 메오리 주소 저장
    printf("%p\n", ptr);        //ptr에 저장된 num의 메모리 주소 출력
    printf("%p\n", &num);       //num의 메모리 주소 출력
    return 0;
}
```

포인터(2) - 역참조1

mic_0315 > test2.c > main()

```c
#include<stdio.h>
int main(){
    int *ptr;                   //포인터 변수 선언
    int num = 10;               //int형 변수를 선언하고 10저장
    ptr = &num;                 //ptr에 num의 메모리 주소 저장
    printf("%d\n",*ptr);        //역참조 연산자로 num의 메모리 주소에 접근하여 값을 가져옴
    return 0;
}
```

포인터(3) - 역참조2

mic_0315 > C test3.c > main()

```c
#include <stdio.h>
int main(){
    int *ptr;                   //포인터 변수 선언
    int num = 10;               //int형 변수를 선언하고 10저장
    ptr = &num;                 //ptr에 num의 메모리 주소 저장
    *ptr = 20;                  //역참조 연산자로 메모리 주소에 접근하여 20을 저장
    printf("%d\n", *ptr);       //역참조 연산자로 메모리 주소에 접근하여 20을 출력
    printf("%d\n", num);        //실제 num의 값도 20으로 바뀌어 출력
    return 0;
}
```

포인터(4) - 이중포인터

mic_0315 > C test4.c > main()

```c
#include <stdio.h>
int main(){
    int *ptr1;              // 포인터 선언
    int **ptr2;             // 이중 포인터 선언
    int num = 10;           // 변수 선언
    ptr1 = &num;            //num의 메모리 주소를 ptr1에 저장
    ptr2 = &ptr1;           //ptr1의 메모리 주소를 ptr2에 저장
    printf("%d\n", **ptr2); //포인터를 두 번 역참조하여 num의 메모리 주소에 접근하여 10출력
    return 0;
}
```

배열과 포인터(2)

mic_0315 > test5.c > main()

```c
#include <stdio.h>
int main(){
    int arr[3] = {0,1,2};   //배열 초기값 출력 해보기
    printf("arr[0]:%d\n", arr[0]);
    printf("arr[1]:%d\n", arr[1]);
    printf("arr[2]:%d\n", arr[2]);
    int* ptr = &arr[0];     //배열을 조작할 포인터 선언, arr[0]첫번째 주소를 가리킴
    *ptr = 10;              //첫번째 값 변경
    *(ptr + 1) = 30;        //두번째 값 변경, 자료형이 int이므로 1씩 증가하면 4byte씩 이동
    *(ptr + 2) = 300;       //세번째 값 변경
    printf("변경 후 배열 출력");    //배열 출력
    printf("arr[0]:%d\n", arr[0]);
    printf("arr[1]:%d\n", arr[1]);
    printf("arr[2]:%d\n", arr[2]);
    return 0;
}
```