

# 운영체제 Assignmnet03

2021270660 이지원

# System Description

```
// 현재 시간을 출력하는 함수
void printCurTime() {
    SYSTEMTIME lt;
    // GetSystem() -> UTC 기준... 따라서 GetLocalTime()을 사용해 시간을 가져옴
    GetLocalTime(&lt); // get local time
    cout << lt.wYear << "-";
    cout.width(2);
    cout.fill('0');
    cout << lt.wMonth << "-";
    cout.width(2);
    cout << lt.wDay << " ";
    cout.width(2);
    cout << lt.wHour << ":";
    cout.width(2);
    cout << lt.wMinute << ":";
    cout.width(2);
    cout << lt.wSecond << " -> ";
}
```

```
// 현재 CPU 부하를 반환하는 함수
double cpu() {
    // 함수 사용을 위한 파라미터
    FILETIME idle, kernel, user;
    double r = 0;

    // t0에서의 CPU 사용 시간 구해오기
    GetSystemTimes(&idle, &kernel, &user);
    DWORD kernelHigh1 = kernel.dwHighDateTime;
    DWORD kernelLow1 = kernel.dwLowDateTime;
    DWORD userHigh1 = user.dwHighDateTime;
    DWORD userLow1 = user.dwLowDateTime;
    DWORD idleHigh1 = idle.dwHighDateTime;
    DWORD idleLow1 = idle.dwLowDateTime;

    Sleep(1000); // 1초 후

    // t1에서의 CPU 사용 시간 구해오기
    GetSystemTimes(&idle, &kernel, &user);
    DWORD kernelHigh2 = kernel.dwHighDateTime;
    DWORD kernelLow2 = kernel.dwLowDateTime;
    DWORD userHigh2 = user.dwHighDateTime;
    DWORD userLow2 = user.dwLowDateTime;
    DWORD idleHigh2 = idle.dwHighDateTime;
    DWORD idleLow2 = idle.dwLowDateTime;

    // 시간 계산
    // 실제 cpu 사용 시간 / 전체 시간 * 100
    double kernelTime = (kernelHigh2 + kernelLow2) - (kernelHigh1 + kernelLow1);
    double userTime = (userHigh2 + userLow2) - (userHigh1 + userLow1);
    double idleTime = (idleHigh2 + idleLow2) - (idleHigh1 + idleLow1);

    double delta = ((userTime + kernelTime) - idleTime) * 100 / (kernelTime + userTime);

    return delta;
}
```

# System Description

```
double getAvgCPU(double total[], int n, int k) {
    double sum = 0;

    if (n == 15) { // 처음부터 전체에 대한 평균일 경우
        for (int i = 0; i < n; i++) {
            sum += total[i];
        }
    } else if (k < 14) { // 15개 미만의 데이터 존재, 최근 10개나 5개 ...
        for (int i = 0; i < n; i++) {
            sum += total[k - i];
        }
    } else { // 전체 중에 최근 10개, 5개만 평균
        for (int i = 0; i < n; i++) {
            int index = k % 15 - i;
            if (index < 0) {
                index = i;
            }
            sum += total[index];
        }
    }
    return sum / n;
}
```

```
void printNumCPU() {
    SYSTEM_INFO info;
    GetSystemInfo(&info);
    cout << "Number of CPU's : " << info.dwNumberOfProcessors << endl;
}
```

# System Description

```
int main(void) {
    double total[15] = { 0, };
    int i = 0;
    int k = 0;

    // 프로세스 개수 출력
    printNumCPU();

    while (true) {
        if (i > 14) {
            k = i%15;
        }
        else {
            k = i;
        }

        total[k] = cpu();

        // 시간 출력
        cout << i << " ";
        printCurTime();

        // 현재 CPU Load 출력
        cout << "[CPU Load : " << showpoint << setprecision(4) << total[k] << "%]";

        // 평균 CPU Load 출력
        if ( i >= 4 ) {
            cout << " [5sec Avg : " << showpoint << setprecision(4) << getAvgCPU(total, 5, i) << "%]";
        }
        if ( i >= 9 ) {
            cout << " [10sec Avg : " << showpoint << setprecision(4) << getAvgCPU(total, 10, i) << "%]";
        }
        if ( i >= 14 ) {
            cout << " [15sec Avg : " << showpoint << setprecision(4) << getAvgCPU(total, 15, i) << "%]";
        }

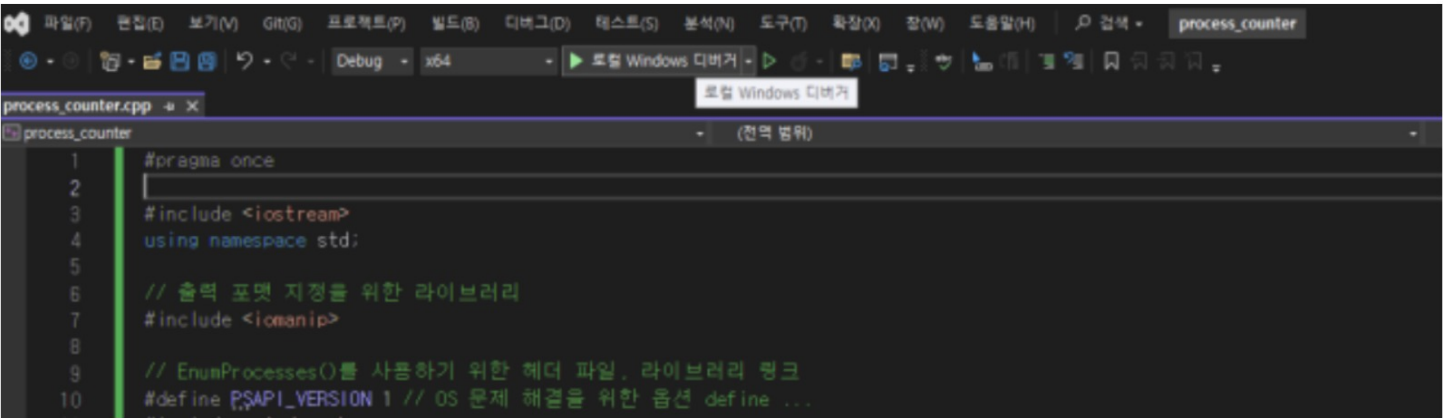
        cout << endl;
        i++;
    }
}
```

# Test Description

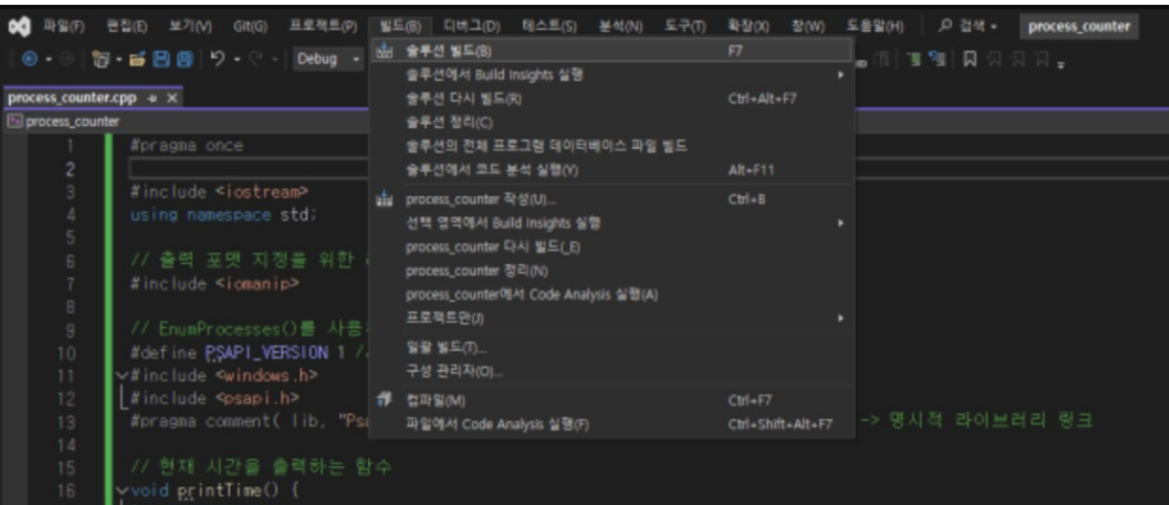
```
C:\Users\WrogerW\source\Wrep\ x + v
Number of CPU's : 8
0 2024-05-29 23:07:58 -> [CPU Load : 1.923%]
1 2024-05-29 23:07:59 -> [CPU Load : 2.734%]
2 2024-05-29 23:08:00 -> [CPU Load : 5.192%]
3 2024-05-29 23:08:01 -> [CPU Load : 4.305%]
4 2024-05-29 23:08:02 -> [CPU Load : 4.031%] [5sec Avg : 3.637%]
5 2024-05-29 23:08:03 -> [CPU Load : 3.314%] [5sec Avg : 3.915%]
6 2024-05-29 23:08:04 -> [CPU Load : 2.930%] [5sec Avg : 3.954%]
7 2024-05-29 23:08:05 -> [CPU Load : 1.923%] [5sec Avg : 3.301%]
8 2024-05-29 23:08:06 -> [CPU Load : 1.734%] [5sec Avg : 2.786%]
9 2024-05-29 23:08:07 -> [CPU Load : 1.536%] [5sec Avg : 2.287%] [10sec Avg : 2.962%]
10 2024-05-29 23:08:08 -> [CPU Load : 1.172%] [5sec Avg : 1.859%] [10sec Avg : 2.887%]
11 2024-05-29 23:08:09 -> [CPU Load : 1.731%] [5sec Avg : 1.619%] [10sec Avg : 2.787%]
12 2024-05-29 23:08:10 -> [CPU Load : 1.172%] [5sec Avg : 1.469%] [10sec Avg : 2.385%]
13 2024-05-29 23:08:11 -> [CPU Load : 5.192%] [5sec Avg : 2.160%] [10sec Avg : 2.473%]
14 2024-05-29 23:08:12 -> [CPU Load : 1.559%] [5sec Avg : 2.165%] [10sec Avg : 2.226%] [15sec Avg : 2.697%]
15 2024-05-29 23:08:13 -> [CPU Load : 6.214%] [5sec Avg : 4.495%] [10sec Avg : 3.391%] [15sec Avg : 2.983%]
16 2024-05-29 23:08:14 -> [CPU Load : 1.758%] [5sec Avg : 4.300%] [10sec Avg : 3.294%] [15sec Avg : 2.917%]
17 2024-05-29 23:08:15 -> [CPU Load : 5.664%] [5sec Avg : 4.394%] [10sec Avg : 3.341%] [15sec Avg : 2.949%]
18 2024-05-29 23:08:16 -> [CPU Load : 2.885%] [5sec Avg : 4.110%] [10sec Avg : 3.199%] [15sec Avg : 2.854%]
19 2024-05-29 23:08:17 -> [CPU Load : 2.539%] [5sec Avg : 3.812%] [10sec Avg : 3.050%] [15sec Avg : 2.755%]
20 2024-05-29 23:08:18 -> [CPU Load : 4.078%] [5sec Avg : 3.385%] [10sec Avg : 3.126%] [15sec Avg : 2.806%]
21 2024-05-29 23:08:19 -> [CPU Load : 1.541%] [5sec Avg : 3.341%] [10sec Avg : 2.987%] [15sec Avg : 2.713%]
22 2024-05-29 23:08:20 -> [CPU Load : 2.148%] [5sec Avg : 2.638%] [10sec Avg : 3.010%] [15sec Avg : 2.728%]
23 2024-05-29 23:08:21 -> [CPU Load : 2.344%] [5sec Avg : 2.530%] [10sec Avg : 3.071%] [15sec Avg : 2.769%]
24 2024-05-29 23:08:22 -> [CPU Load : 1.923%] [5sec Avg : 2.407%] [10sec Avg : 3.109%] [15sec Avg : 2.795%]
25 2024-05-29 23:08:23 -> [CPU Load : 3.320%] [5sec Avg : 2.255%] [10sec Avg : 2.820%] [15sec Avg : 2.938%]
26 2024-05-29 23:08:24 -> [CPU Load : 1.346%] [5sec Avg : 2.216%] [10sec Avg : 2.779%] [15sec Avg : 2.912%]
27 2024-05-29 23:08:25 -> [CPU Load : 0.9615%] [5sec Avg : 1.979%] [10sec Avg : 2.309%] [15sec Avg : 2.898%]
28 2024-05-29 23:08:26 -> [CPU Load : 1.562%] [5sec Avg : 1.823%] [10sec Avg : 2.176%] [15sec Avg : 2.656%]
```

# User Documentation

- 1. .exe 파일 실행 ( 빌드된 상태 )
- 2. 로컬 디버거로 실행 : 프로젝트 생성 후 소스 파일에 추가 후 로컬 디버거로 실행



- 3. 빌드 후 .exe 파일 실행 : 마찬가지로 프로젝트 생성 후 빌드 - 솔루션 빌드 후 .exe 파일을 찾아 실행



user > source > repos > process_counter > x64 > Debug				
이름	수정한 날짜	유형	크기	
process_counter	2024-04-08 오후 5:08	응용 프로그램	73KB	
process_counter.pdb	2024-04-08 오후 5:08	Program Debug ...	1,348KB	