

Software Portfolio

한승환

- 성인 동영상 탐색 프로그램

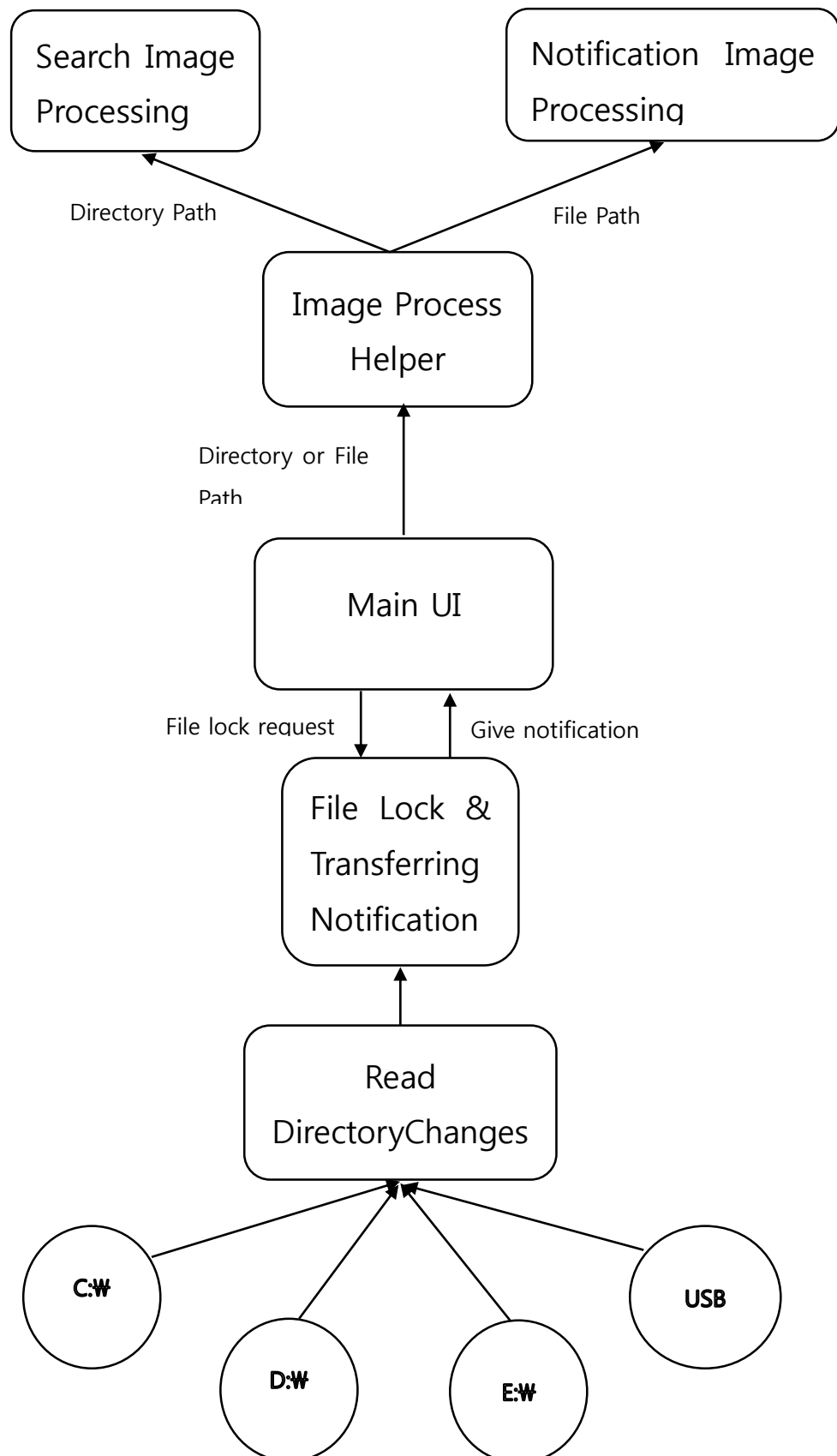
1. 기능

- 하드디스크 내에 존재하는 야동을 찾아냄
- 실시간 감시 기능 활성화로 전체 디스크 내에서 동영상 파일 추가 감시 및 자동 탐지
- 야동으로 판단된 동영상 실행 불가
- USB 삽입 시 USB 자동 검사 및 실시간 감시
- 자가 프로세스 보호(작업관리자 -> 프로세스 강제 종료 방지)

2. 목적

- 과장되고 잘못된 상식이 담겨있는 야동을 제한함으로써 학습률이 높은 청소년들의 잘못된 성지식 습득을 방지하고자 만듦

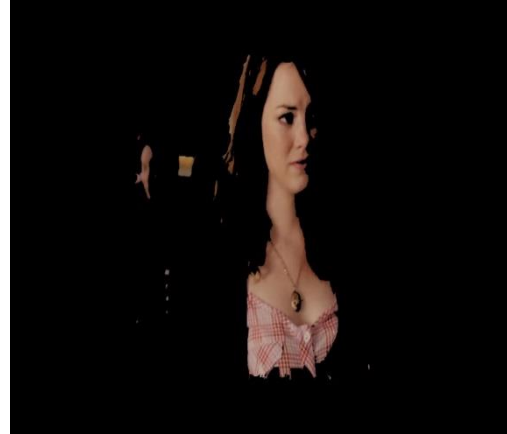
3. 프로그램 구조



4. 기능 구현 방법

- 성인 동영상 탐지 원리

1. Grabcut 알고리즘 사용(OpenCV) - 사람 or 물체 구별



```
GrabcutHelper helper;
bool isFound = false;

helper.setImage(frame, winName);
helper.setRect();

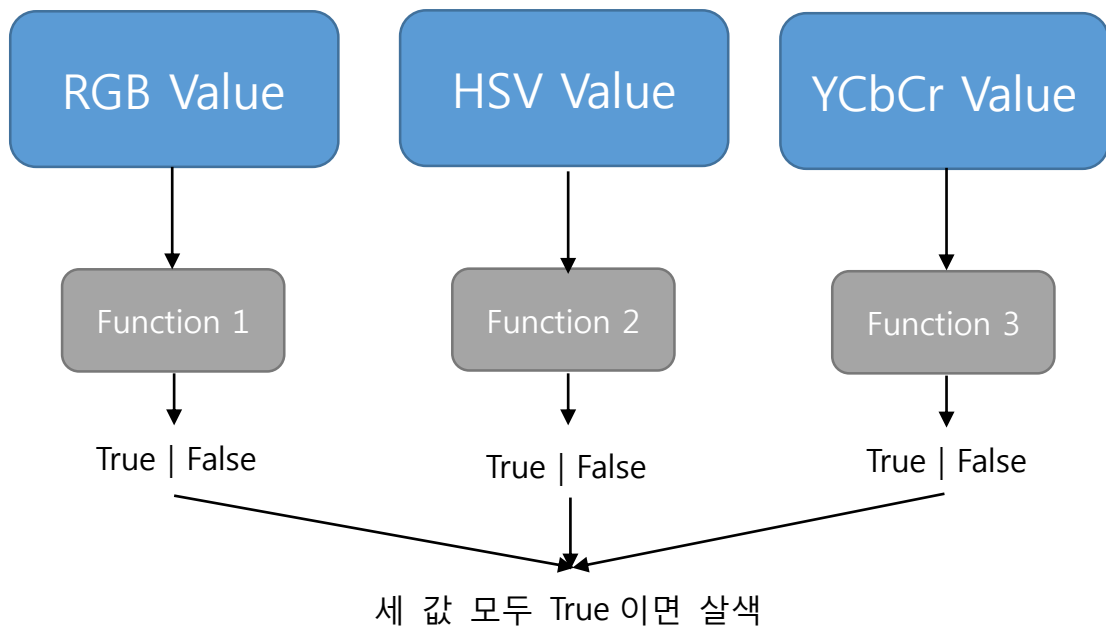
int iterCount = helper.getIterCount();
int newIterCount = helper.doGrabcut();
if (newIterCount > iterCount)
    helper.setTransformedImage();

int GrabcutHelper::doGrabcut(){
    if (isInitialized)
        grabCut(*image, mask, rect, bgdModel, fgdModel, 1);
    else{
        if (rectState != true)
            return iterCount;

        grabCut(*image, mask, rect, bgdModel, fgdModel, 1, GC_INIT_WITH_RECT);
        isInitialized = true;
    }
    iterCount++;

    return iterCount;
}
```

2. 피부색 판별 알고리즘 - 다른 색 표현 방식으로 변환 후 결과 값 비교(HSV, YCbCr)



Ex)



```

bool R1(int R, int G, int B) {
    bool e1 = (R>95) && (G>40) && (B>20) && ((max(R, max(G, B)) - min(R, min(G, B)))>15) && (abs(R - G)>15) && (R>G) && (R>B);
    bool e2 = (R>220) && (G>210) && (B>170) && (abs(R - G) <= 15) && (R>B) && (G>B);
    return (e1 || e2);
}

bool R2(float Y, float Cr, float Cb) {
    bool e3 = Cr <= 1.5862*Cb + 20;
    bool e4 = Cr >= 0.3448*Cb + 76.2069;
    bool e5 = Cr >= -4.5652*Cb + 234.5652;
    bool e6 = Cr <= -1.15*Cb + 301.75;
    bool e7 = Cr <= -2.2857*Cb + 432.85;
    return e3 && e4 && e5 && e6 && e7;
}

bool R3(float H, float S, float V) {
    return (H<25) || (H > 230);
}

```

```

for (int i = 0; i < src.rows; i++) {
    for (int j = 0; j < src.cols; j++) {

        Vec3b pix_bgr = src.ptr<Vec3b>(i)[j];
        int B = pix_bgr.val[0];
        int G = pix_bgr.val[1];
        int R = pix_bgr.val[2];
        // apply rgb rule
        bool a = R1(R, G, B);

        Vec3b pix_ycrCb = src_ycrCb.ptr<Vec3b>(i)[j];
        int Y = pix_ycrCb.val[0];
        int Cr = pix_ycrCb.val[1];
        int Cb = pix_ycrCb.val[2];
        // apply ycrCb rule
        bool b = R2(Y, Cr, Cb);

        Vec3f pix_hsv = src_hsv.ptr<Vec3f>(i)[j];
        float H = pix_hsv.val[0];
        float S = pix_hsv.val[1];
        float V = pix_hsv.val[2];
        // apply hsv rule
        bool c = R3(H, S, V);

        if (!(a&&b&&c))
            dst.ptr<Vec3b>(i)[j] = cblack;
    }
}

```

```

Mat tempImage = helper.getConvertedImage();
int totalCount = getCount(tempImage);

Mat skin = GetSkin(tempImage);
int skinCount = getCount(skin);

double skinRatio = (double)((double)skinCount / (double)totalCount);

if (skinRatio >= pInfo->choiceLevel)
{
    isFound = true;
    break;
}

```

- 실시간 감시 및 파일 실행 방지

1. 실시간 감시 - ReadDirectoryChangesW API 사용

Retrieves information that describes the changes within the specified directory.

```
BOOL WINAPI ReadDirectoryChangesW(  
    _In_      HANDLE          hDirectory,  
    _Out_     LPVOID          lpBuffer,  
    _In_      DWORD           nBufferLength,  
    _In_      BOOL            bWatchSubtree,  
    _In_      DWORD           dwNotifyFilter,  
    _Out_opt_ LPDWORD          lpBytesReturned,  
    _Inout_opt_ LPOVERLAPPED  lpOverlapped,  
    _In_opt_   LPOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine  
);
```

```
DWORD dwBytes = 0;  
  
// This call needs to be reissued after every APC.  
BOOL success = ::ReadDirectoryChangesW(  
    m_hDirectory,          // handle to directory  
    &m_Buffer[0],          // read results buffer  
    m_Buffer.size(),      // length of buffer  
    m_bChildren,          // monitoring option  
    FILE_NOTIFY_CHANGE_FILE_NAME, // filter conditions  
    &dwBytes,             // bytes returned  
    &m_Overlapped,        // overlapped buffer  
    &NotificationCompletion); // completion routine  
  
VOID CALLBACK CReadChangesRequest::NotificationCompletion(  
    DWORD dwErrorCode,          // completion code  
    DWORD dwNumberOfBytesTransferred, // number of bytes transferred  
    LPOVERLAPPED lpOverlapped) // I/O information buffer  
{  
    CReadChangesRequest* pBlock = (CReadChangesRequest*)lpOverlapped->hEvent;  
  
    if (dwErrorCode == ERROR_OPERATION_ABORTED)  
    {  
        ::InterlockedDecrement(&pBlock->m_pServer->m_nOutstandingRequests);  
        delete pBlock;  
        return;  
    }  
  
    // Can't use sizeof(FILE_NOTIFY_INFORMATION) because  
    // the structure is padded to 16 bytes.  
    _ASSERT(dwNumberOfBytesTransferred >= offsetof(FILE_NOTIFY_INFORMATION, FileName) + sizeof(WCHAR));  
  
    // This might mean overflow? Not sure.  
    if (!dwNumberOfBytesTransferred)  
        return;  
  
    pBlock->BackupBuffer(dwNumberOfBytesTransferred);  
  
    // Get the new read issued as fast as possible. The documentation  
    // says that the original OVERLAPPED structure will not be used  
    // again once the completion routine is called.  
    pBlock->BeginRead();  
  
    pBlock->ProcessNotification();  
}
```

-> 감시 대상인 디렉토리 내에서 파일 이동&추가 이벤트가 생겼을 때, Completion Routine이 호출되고, 그 Completion Routine에서 일련의 작업이 행해진 후 다시 ReadDirectoryChanges 적용

2. 파일 잠금

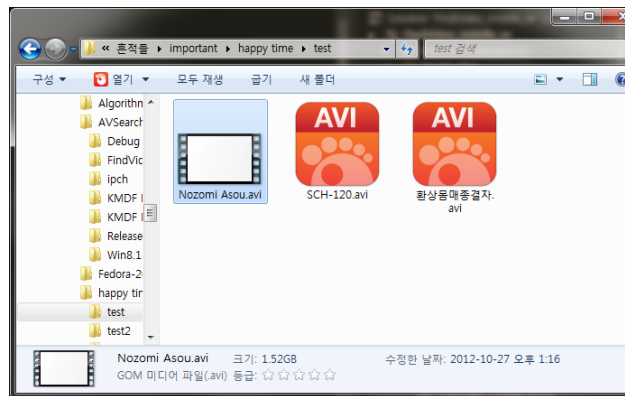
Locks the specified file for exclusive access by the calling process.

```
BOOL WINAPI LockFile(  
    _In_ HANDLE hFile,  
    _In_ DWORD dwFileOffsetLow,  
    _In_ DWORD dwFileOffsetHigh,  
    _In_ DWORD nNumberOfBytesToLockLow,  
    _In_ DWORD nNumberOfBytesToLockHigh  
);
```

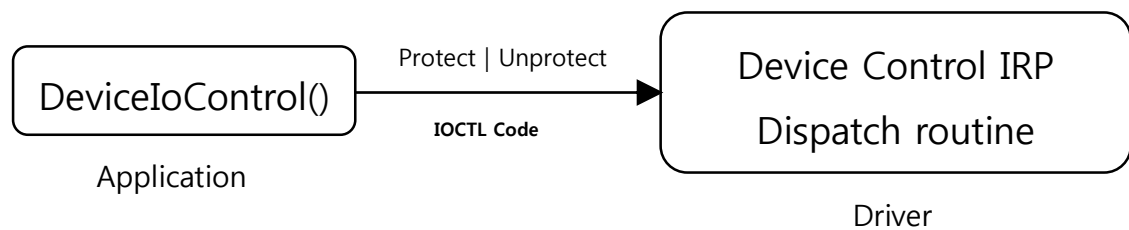
```
BOOL FileLockThread::MyFileLock()  
{  
    for (auto it = fileHandles.begin(); it != fileHandles.end(); it++)  
        LockFile(*it, 0, 0, GetFileSize(*it, 0), 0);  
    return TRUE;  
}
```

-> 성인 동영상이라고 판단된 동영상 파일의 Handle을 얻어서, 동영상 파일 offset과 Size와 함께 LockFile 인자로 넘겨줌

Ex)



- 자가 프로세스 보호 : Kerner driver 이용



-> Application과 Driver 간 IOCTL를 이용한 통신

Sends a control code directly to a specified device driver, causing the corresponding device to perform the corresponding operation.

Syntax

C++

```
BOOL WINAPI DeviceIoControl(
    _In_      HANDLE      hDevice,
    _In_      DWORD       dwIoControlCode,
    _In_opt_  LPVOID      lpInBuffer,
    _In_      DWORD       nInBufferSize,
    _Out_opt_ LPVOID      lpOutBuffer,
    _In_      DWORD       nOutBufferSize,
    _Out_opt_ LPDWORD     lpBytesReturned,
    _Inout_opt_ LPOVERLAPPED lpOverlapped
);
```

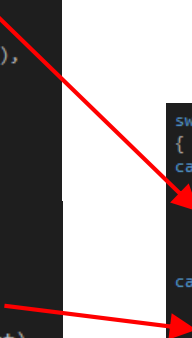
이 API를 이용해서 커널 드라이버와 통신함

```
Result = DeviceIoControl (
    TcDeviceHandle,
    TD_IOCTL_PROTECT_NAME_CALLBACK,
    &ProtectNameCallbackInput,
    sizeof(ProtectNameCallbackInput),
    NULL,
    0,
    &BytesReturned,
    NULL
);

BOOL Result = DeviceIoControl (
    TcDeviceHandle,
    TD_IOCTL_UNPROTECT_CALLBACK,
    &UnprotectCallbackInput,
    sizeof(UnprotectCallbackInput),
    NULL,
    0,
    &BytesReturned,
    NULL
);
```

```
switch (Ioctl)
{
    case TD_IOCTL_PROTECT_NAME_CALLBACK:
        Status = TdControlProtectName (DeviceObject, Irp);
        break;

    case TD_IOCTL_UNPROTECT_CALLBACK:
        Status = TdControlUnprotect (DeviceObject, Irp);
        break;
}
```



The **ObRegisterCallbacks** routine registers a list of callback routines for thread, process, and desktop handle operations.

Syntax

C++

```
NTSTATUS ObRegisterCallbacks(  
    _In_ POB_CALLBACK_REGISTRATION CallbackRegistration,  
    _Out_ PVOID *RegistrationHandle  
);
```

Parameters

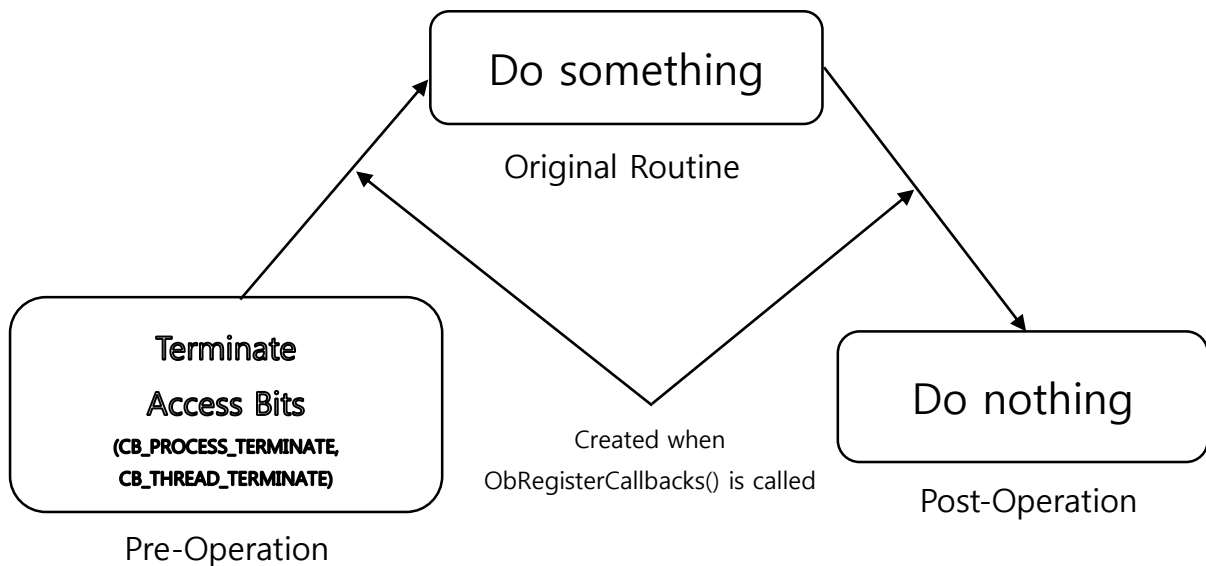
CallbackRegistration [in]

A pointer to an **OB_CALLBACK_REGISTRATION** structure that specifies the list of callback routines and other registration information.

RegistrationHandle [out]

A pointer to a variable that receives a value that identifies the set of registered callback routines. The caller passes this value to the **ObUnRegisterCallbacks** routine to unregister the set of callbacks.

-> 그리고 이 Kernel-driver API로 프로세스 & 스레드 작업에 대한 Filtering routine 생성



```

CBOperationRegistrations[0].ObjectType = PsProcessType;
CBOperationRegistrations[0].Operations |= OB_OPERATION_HANDLE_CREATE;
CBOperationRegistrations[0].Operations |= OB_OPERATION_HANDLE_DUPLICATE;
CBOperationRegistrations[0].PreOperation = CBTdPreOperationCallback;
CBOperationRegistrations[0].PostOperation = CBTdPostOperationCallback;

CBOperationRegistrations[1].ObjectType = PsThreadType;
CBOperationRegistrations[1].Operations |= OB_OPERATION_HANDLE_CREATE;
CBOperationRegistrations[1].Operations |= OB_OPERATION_HANDLE_DUPLICATE;
CBOperationRegistrations[1].PreOperation = CBTdPreOperationCallback;
CBOperationRegistrations[1].PostOperation = CBTdPostOperationCallback;

RtlInitUnicodeString (&CBAltitude, L"1000");

CBObRegistration.Version = OB_FLT_REGISTRATION_VERSION;
CBObRegistration.OperationRegistrationCount = 2;
CBObRegistration.Altitude = CBAltitude;
CBObRegistration.RegistrationContext = &CBCallbackRegistration;
CBObRegistration.OperationRegistration = CBOperationRegistrations;

Status = ObRegisterCallbacks (
    &CBObRegistration,
    &pCBRegistrationHandle
);

```

-> Pre-Operation & Post-Operation 함수 등록

```

OB_PREOP_CALLBACK_STATUS
CBTdPreOperationCallback (
    _In_ PVOID RegistrationContext,
    _Inout_ POB_PRE_OPERATION_INFORMATION PreInfo
)

```

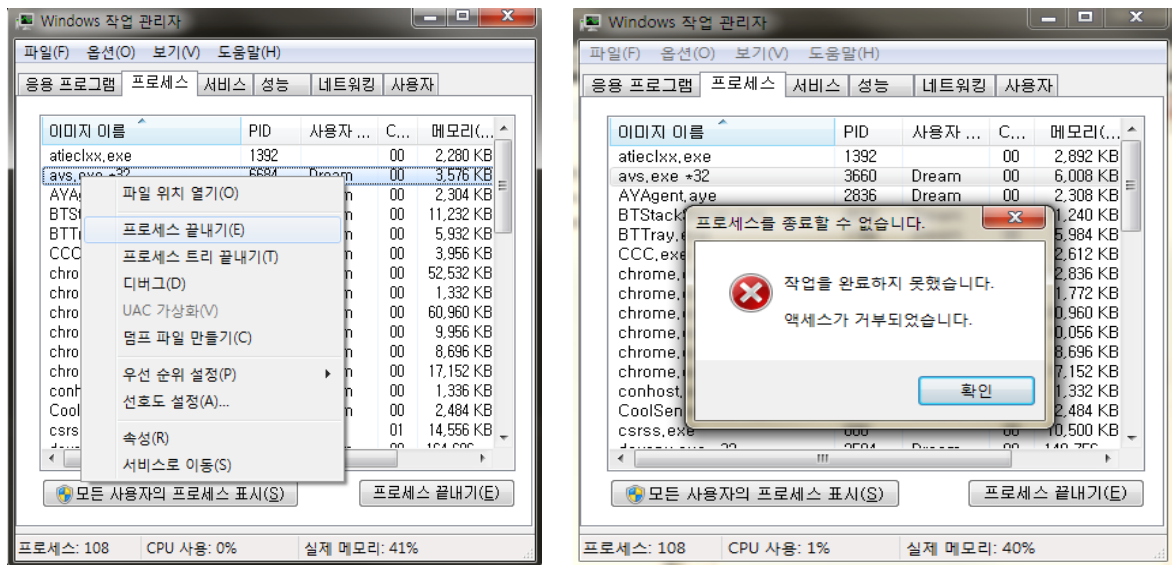
```

// Filter only if request made outside of the kernel
if (PreInfo->KernelHandle != 1) {
    *DesiredAccess &= ~AccessBitsToClear;
    *DesiredAccess |= AccessBitsToSet;
}

```

-> Pre-Operation Routine에서 Process나 Thread에 대한 작업을 하기 위한 handle 을 얻을 때, 목표 프로세스에 대한 종료 권한을 삭제

Ex)



-> 관리자가 설정한 비밀번호로만 프로그램을 종료할 수 있기 때문에, 감시 프로그램으로서의 모습을 갖추(비밀번호는 SHA-1로 저장되어있어 사실상 원문을 알아내는 건 불가능)

5. 프로그램 모습

