

---

# **Calling Convention**

## **Code Cave**

### **Lena tutorial #15**

# Calling Convention

(함수 호출 규칙)

함수를 호출할 때 인자를  
전달하는 방식이나 함수  
실행이 끝나고 스택에  
정리하는 방식에 대한 약속

`cdecl`

`stdcall`

`fastcall`

# Calling Convention

(함수 호출 규칙)

>고급언어

```
void caller() {  
    callee(1,2);  
    return;  
}
```



[caller]  
**PUSH 2**  
**PUSH 1**  
CALL callee  
**ADD ESP 8**

인자 전달 방법 : 스택  
인자 전달 순서 : 오른 -> 왼  
스택 정리 : 호출자 -> 피호출자

[callee]  
X

**cdecl**

stdcall

fastcall

# Calling Convention

(함수 호출 규약)

>고급언어

```
void caller() {
    callee(1,2);
    return;
}
```

인자 전달 방법 : 스택  
인자 전달 순서 : 오른 -> 왼  
스택 정리 : 피호출자



[caller]  
**PUSH 2**  
**PUSH 1**  
CALL callee  
.....

[callee]  
PUSH EBP  
MOV EBP, ESP  
.....  
POP EBP  
**RETN 8**

cdecl

**stdcall**

fastcall

# Calling Convention

(함수 호출 규약)

>고급언어

```
void caller() {
    callee(1,2);
    return;
}
```



```
[caller]
MOV EDX 2
MOV ECX 1
CALL callee
.....
```

인자 전달 방법 : 레지스터, 스택  
 인자 전달 순서 : ECX, EDX 레지스터로  
 전달 후 스택 (오른 -> 왼)  
 스택 정리 : X (레지스터를 사용하기 때  
 문)

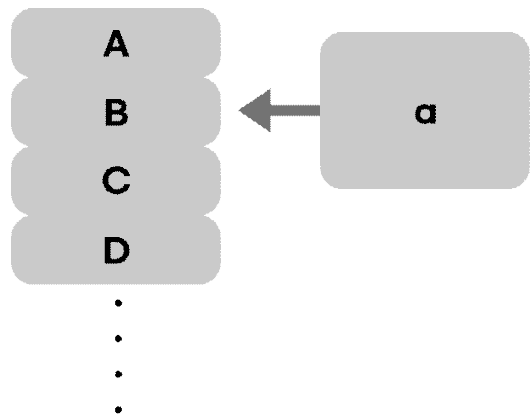
```
[callee]
PUSH EBP
MOV EBP, ESP
.....,
MOV DWORD PTR SS:[EBP-8],EDX
MOV DWORD PTR SS:[EBP-4],ECX
.....,
POP EBP
RETN
```

cdecl

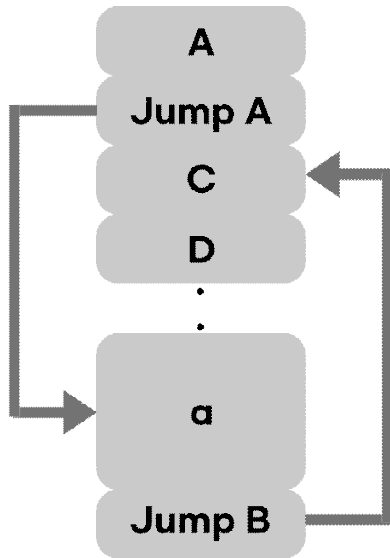
stdcall

fastcall

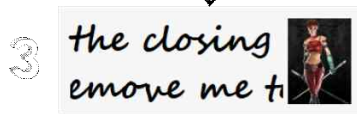
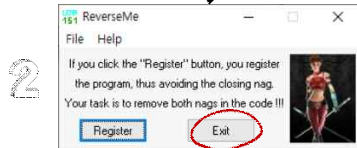
# Code Cave (코드 케이브)



새로운 코드 크기 < 기존 코드 크기 (O)  
새로운 코드 크기 > 기존 코드 크기 (X)



# Lena Tutorial #15



목표 : Nag 창 없애기

1, 3 - Nag 창  
2 - 정상 창

# Lena Tutorial #15

## Call stack window : 콜 스택들을 모아서 보여줌

정상

Call stack of main thread

Stack	Data	Procedure	Called From	Frame
0012BF0C	750CCDE0	ntdll.KiFastSystemCall	USER32.750CCDE0	0012BF30
0012BF10	750C18D9	USER32.750CC004	USER32.GetMessageA+3B	
0012BF34	6F2F539B	USER32.GetMessageA	MFC42.#5307+10	
0012BF50	6F30E81A	MFC42.#5307	MFC42.#5718+000	0012BF6C
0012BF70	6F321640	MFC42.#5718	MFC42.#2514+110	0012C004
0012BF84	0042039F	MFC42.#2514	ReverseMe_NAGS.0042039A	0012BF80
0012FBCC	0041D57E	ReverseMe_NAGS.00420050	ReverseMe_NAGS.0041D579	01160014
0012FA10	0041A9F9	ReverseMe_NAGS.0041D090	ReverseMe_NAGS.0041A9F4	
0012FA8C	0041A81F	ReverseMe_NAGS.0041A8D0	ReverseMe_NAGS.0041A81A	0012FA88
0012FADC	0041A9A7	ReverseMe_NAGS.0041A7C0	ReverseMe_NAGS.0041A9A2	0012FAD8
0012FB58	0041A81F	ReverseMe_NAGS.0041A8D0	ReverseMe_NAGS.0041A81A	0012FB54
0012FB88	0042C8F9	ReverseMe_NAGS.0041A7C0	ReverseMe_NAGS.0042C8F4	0012FB84
0012FC08	6F2F3A93	ReverseMe_NAGS.0042C1E0	MFC42.#1576+0	0012FC04
0012FEDC	00433705	MFC42.#1576	ReverseMe_NAGS.00433700	0012FED8
0012FEF0	004332EC	ReverseMe_NAGS.004336F0	ReverseMe_NAGS.<ModuleEntry>	0012FE88
0012FF8C	75A73C45	???	kernel32.BaseThreadInitThunk	
0012FF98	76EB37F5	???	ntdll.76EB37F3	0012FF94
0012FFD8	76EB37C8	ntdll.76EB37CE	ntdll.76EB37C3	0012FFD4

?

Call stack of main thread

Stack	Data	Procedure	Called From
0019BED0	76B01A95	USER32.NtUserGetMessage	USER32.GetMessageA+4F
0019BF10	51A84425	MFC42.#5307	MFC42.#5307
0019BF2C	51ACE253	MFC42.#5307	MFC42.#5718+101
0019BF54	51AE0ADD	MFC42.#5718	MFC42.#2514+0E8

stack

0019BED0	76B01A95	RETURN from win32u.NtUserGetMessage to USER32.GetMessageA+55
0019BED4	004467CC	Igd Arg1 = ReverseMe_NAGS.4467CC
0019BED8	00000000	Arg2 = 0
0019BEDC	00000000	Arg3 = 0
0019BEE0	00000000	Arg4 =
0019BEE4	004467CC	Igd
0019BEE8	00432020	-C Jump to MFC42.#5307
0019BEEC	51B7F04C	LQ-Q
0019BEF0	0019BF20	z-
0019BEF4	51A84425	NO-Q RETURN from MFC42.51A84425 to MFC42.#1168+0B4
0019BEF8	D38AEC3E	>150
0019BEFC	0019BFEC	1z- PTR to ASCII "-,C"
0019BF00	0019BF1C	z-
0019BF04	51A4AC9D	LQ-Q RETURN from ntdll.RtlLeaveCriticalSection to USER32.#3030+5D
0019BF08	51B7F04C	LQ-Q
0019BF0C	0019BF50	Pz-
0019BF10	51A84425	NO-Q RETURN from USER32.GetMessageA to MFC42.#5307+15
0019BF14	004467CC	Igd pMsg = ReverseMe_NAGS.004467CC -> MSGA {hwnd=NULL, Msg=WM_TIMER}
0019BF18	00000000	hwnd = NULL
0019BF1C	00000000	MsgFilterWin = 0
0019BF20	00000000	MsgFilterMax =
0019BF24	0019BFEC	1z- PTR to ASCII "-,C"
0019BF28	00446798	gd PTR to ASCII "B/C"
0019BF2C	51ACE253	SA-Q RETURN to MFC42.#5718+103



# Lena Tutorial #15

## stack

```

0019BED0 76B01A95 --v RETURN from win32u.NtUserGetMessage to USER32.GetMessageA+55
0019BED4 004467CC IgD Arg1 = ReverseMe__NAGs.4467CC
0019BED8 00000000 Arg2 = 0
0019BEDC 00000000 Arg3 = 0
0019BEE0 00000000 Arg4 = 0
0019BEE4 004467CC IgD
0019BEE8 00432020 <C
0019BEEC 51B7F04C Lb-Q
0019BEF0 0019BF20 <+
0019BEF4 51AA51A4 RQ+Q RETURN from MFC42.51AB70D6 to MFC42.#1168+084
0019BEF8 D38AEC3E >+SO
0019BEFC 0019BFEC <+ PTR to ASCII "-",C"
0019BF00 0019BF1C <+
0019BF04 51AA4C9D L+Q RETURN from ntdll.RtlLeaveCriticalSection to MFC42.#3030+5D
0019BF08 51B7F04C Lb-Q
0019BF0C 0019BF50 P+
0019BF10 51AB4425 Nd-Q RETURN from USER32.GetMessageA to MFC42.#307+15
0019BF14 004467CC IgD
0019BF18 00000000 hand = NULL
0019BF1C 00000000 MsgFilterMin = 0
0019BF20 00000000 MsgFilterMax = 0
0019BF24 0019BFEC <+ PTR to ASCII "-",C"
0019BF28 00446798 <D PTR to ASCII "B/C"
0019BF2C 51ACE253 S+Q RETURN to MFC42.#5718+103

```

```

0019BF80 00000000
0019BF84 00400000 @ OFFSET ReverseMe__NAGs.<STRUCT IMAGE_DOS_HEADER>
0019BF88 0019BF5C <+
0019BF8C 0019F8A8 <+ Pointer to next SEH record
0019BF90 51A89295 <+Q SE handler
0019BF94 00000000
0019BF98 00A10014 <+
0019BF9C 0043039F RETURN from MFC42.#2514 to ReverseMe__NAGs.0042039F
0019BFA0 00000000

```

## Code

```

00420351 8942 0C MOV DWORD PTR DS:[EDX+0C],EAX
00420354 66B8 7000 MOV AX,7D
00420358 66B8942 10 MOV WORD PTR DS:[EDX+10],AX
0042035C E8 3F47FFFF CALL 00414AA0
00420361 6A 00 PUSH 0
00420363 6A 00 PUSH 0
00420365 53 PUSH EBX
00420366 8D4C24 58 LEA ECX,[ESP+58]
0042036A C68424 1C3900 MOV BYTE PTR SS:[ESP+391C],3
00420372 E8 C7290100 CALL <JMP.MFC42.#3857>
00420377 85C0 TEST EAX,EAX
00420379 74 3F JZ SHORT 0042038A
0042037B 8D4C24 4C LEA ECX,[ESP+4C]
0042037F 89B424 CC3700 MOV DWORD PTR SS:[ESP+37CC],ESI
00420386 890D F4694400 MOV DWORD PTR DS:[4469F4],ECX
0042038C 8B56 08 MOV EDI,DWORD PTR DS:[ESI+8]
0042038F 8D4C24 4C LEA ECX,[ESP+4C]
00420393 899424 A03700 MOV DWORD PTR SS:[ESP+37A0],EDX
0042039A E8 C1280100 CALL <JMP.MFC42.#2514>
0042039F 8D8424 C83700 LEA EAX,[ESP+37C8]
004203A6 8D4C24 20 LEA ECX,[ESP+20]
004203AA 50 PUSH EAX
004203AB E8 2A270100 CALL <JMP.MFC42.#858>
004203B0 8B4C24 1C MOV ECX,DWORD PTR SS:[ESP+1C]
004203B4 890D F4694400 MOV DWORD PTR DS:[4469F4],ECX
004203BA > 55 PUSH EBP
004203BB FF15 7C804300 CALL DWORD PTR DS:[<&KERNEL32.GlobalUnlock>]
004203C1 55 PUSH EBP
004203C2 FF15 80804300 CALL DWORD PTR DS:[<&KERNEL32.GlobalFree>]

```

```

ReverseMe__NAGs.0
Arg3 = 0
Arg2 = 0
Arg1
MFC42.#395
MFC42.#251
Arg1
MFC42.#85
KERNEL32.GlobalUn
KERNEL32.GlobalFr

```

# Lena Tutorial #15

## Code

```

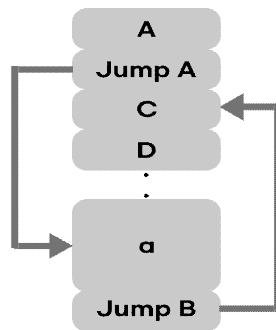
00420377  E9 EE790100 JMP 00437D6A
0042037C  90          NOP
0042037D  90          NOP
0042037E  90          NOP
0042037F  89B424 CC3700 MOV DWORD PTR SS:[ESP+37CC],ESI

```

```

00437D6A  0005 905E4400 ADD BYTE PTR DS:[445E90],AL
00437D70  803D 905E4400 CMP BYTE PTR DS:[445E90],2
00437D77  ^ 0F85 3D86FEFF JNE 004203BA
00437D7D  8D4CE4 4C    LEA ECX,[ESP+4C]
00437D81  ^ E9 F985FEFF JMP 0042037F
00437D86  00          DB 00
00437D87  00          DB 00

```



445E90에 AL값을 더함  
AL 값이 2가 되었을때 창을 보여줌

---

**End**