CSE 376 – Advanced Systems Programming in Unix/C | **Stony Brook University**
Spring 2021 | **Computer Science Department**
February 20, 2021 | **Handout No. 4**

Why You Need Double "int **" Pointers

Enclosed is a small program which shows you why you need to pass `int **` to a function, if you want that function to change a pointer to an integer. There is also the output from the program, and the corresponding symbol table and memory contents for the relevant symbols.

1 The Program

```
/*
 * ptr.c:
 *
 * Test pointers to arrays, and show why it is necessary to
 * pass an int** to the init function if you want to change an array of
 * integers.
 *
 * Erez Zadok.
 *
 */

#include <stdio.h>
#include <malloc.h>

/* This function takes a pointer to an integer, which should not work */
void init1(int *ap, int size)
{
    int *tmp = NULL;
    int i;

    printf("\tap inside init1 is %x\n", (int) ap);

    tmp = (int *) malloc(sizeof(int) * size);
    if (!tmp) {
        fprintf(stderr, "no more memory");
        exit(1);
    }
    printf("\ttmp inside init1 is %x\n", (int) tmp);

    for (i=0; i<size; ++i)      /* zero out the array */
        tmp[i] = 100 + i;
    ap = tmp;

    for (i=0; i<size; ++i)      /* print the array INSIDE init1 */
        printf("\tInside init ap[%d] = %d\n", i, ap[i]);

}

/* This function takes a DOUBLE pointer to an integer, which should work */
void init2(int **app, int size)
{
```

```c
    int *tmp = NULL;
    int i;

    printf("\tapp inside init2 is %x\n", (int) app);
    printf("\t*app inside init2 is %x\n", (int) *app);
    tmp = (int *) malloc(sizeof(int) * size);
    if (!tmp) {
        fprintf(stderr, "no more memory");
        exit(1);
    }

    printf("\ttmp inside init2 is %x\n", (int) tmp);

    for (i=0; i<size; ++i)      /* zero out the array */
        tmp[i] = 200 + i;
    (*app) = tmp;
}

int main()
{
    int *X, *Y, length;
    int i;

    length = 10;
    X = NULL;
    Y = NULL;

    printf("X before init1() is %x\n", (int) X);
    printf("Address of X before init1() is %x\n", (int) &X);
    init1(X, length);
    printf("X after init1() is %x\n", (int) X);
    printf("Address of X after init1() is %x\n", (int) &X);
    if (X == NULL) {
        printf("DID NOT WORK: X is still NULL\n");
    } else {
        printf("Dereferencing *X after init1() is %x\n", (int) (*X));
        for (i=0; i<length; ++i)        /* print the array */
            printf("X[%d] = %d\n", i, X[i]);
    }

    printf("\n");

    printf("Y before init1() is %x\n", (int) Y);
    printf("Address of Y before init1() is %x\n", (int) &Y);
    init2(&Y, length);
    printf("Y after init1() is %x\n", (int) Y);
    printf("Address of Y after init1() is %x\n", (int) &Y);
    if (Y == NULL) {
        printf("DID NOT WORK: Y is still NULL\n");
    } else {
        printf("Dereferencing *Y after init1() is %d\n", (int) (*Y));
        for (i=0; i<length; ++i)        /* print the array */
            printf("Y[%d] = %d\n", i, Y[i]);
    }

    exit(0);
}
```

2 Output from the Program

```
$ ./ptr
X before init1() is 0
Address of X before init1() is effff0a4
        ap inside init1 is 0
        tmp inside init1 is 23110
        Inside init ap[0] = 100
        Inside init ap[1] = 101
        Inside init ap[2] = 102
        Inside init ap[3] = 103
        Inside init ap[4] = 104
        Inside init ap[5] = 105
        Inside init ap[6] = 106
        Inside init ap[7] = 107
        Inside init ap[8] = 108
        Inside init ap[9] = 109
X after init1() is 0
Address of X after init1() is effff0a4
DID NOT WORK: X is still NULL

Y before init1() is 0
Address of Y before init1() is effff0a0
        app inside init2 is effff0a0
        *app inside init2 is 0
        tmp inside init2 is 23140
Y after init1() is 23140
Address of Y after init1() is effff0a0
Dereferencing *Y after init1() is 200
Y[0] = 200
Y[1] = 201
Y[2] = 202
Y[3] = 203
Y[4] = 204
Y[5] = 205
Y[6] = 206
Y[7] = 207
Y[8] = 208
Y[9] = 209
```

3 The Symbol Table

Symbol	**Address**
X	effff0a4
Y	effff0a0

4 Memory Contents

Memory Address	**Contents Before** `init`	**Contents After** `init`
effff0a4	0	0
effff0a0	0	23140