22/10/2016

Αναφορά Εργαστηρίου 1

Ομάδα LAB20130164

Μανώλης Πετράκος
Δημήτρης Καραμπάσογλου

Προεργασία

Ο Σκοπός της άσκησης είναι η εξοικείωση με τον τρόπο που αποθηκεύει η C τις διάφορες μεταβλητές και η εξοικείωση με την δεκαεξαδική αναπαράσταση τους. Προαπαιτούμενα είναι η γνώση της C και των δεικτών διεύθυνσης όπως και η γνώση του τρόπου που αποθηκεύονται οι διάφορες μεταβλητές.

Περιγραφή Ζητούμενων

Στην αρχή έπρεπε να μελετηθούν διάφορες μεταβλητές πως αποθηκεύονται στη μνήμη. Μετά έπρεπε να γράψουμε κώδικα για:

Εκτύπωση διευθύνσεων απευθείας ή με την χρήση δεικτών. Επίσης εκτύπωση μεγέθους ενός πίνακα και ενός στοιχείου του.

Εισαγωγή τριών μεταβλητών και εκτύπωση των αποστάσεων μεταξύ τους.

Δήλωση δομών και εκτύπωση των μεγεθών τους.Δέσμευση μνήμης χρησιμοποιώντας την εντολή malloc() και εκτύπωση των αποστάσεων των δεικτών.

Εμφάνιση διεύθυνσης τριών μεταβλητών, μία global, μια τοπική και μίας malloc(). Επίσης εμφανίζεται και η θέση του κώδικα της main().

Περιγραφής της Εκτέλεσης

Α. Το m αποθηκεύει τη τιμή του και μετά το τέλος της συνάρτησης επειδή είναι static αντίθετα με το n που χάνεται η τιμή του. Το var2 αποθηκεύεται στην επόμενη θέση από το var1 και προσπελάζεται προσθέτοντας 1 στον pointer του var1.

	A	**A+1**	**(int)A+1**	**&A[1]**
decimal:	2686708	2686712	2686709	2686712
hexadecimal:	0x28fef4	0x28fef8	0x28fef5	0x28fef8

Το A+1 και το &A[1] δείχνουν τη θέση του δεύτερου στοιχείου του πίνακα. Το (int)A+1 παίρνει τη τιμή της διεύθυνσης του A και προσθέτει 1.Αυτό γίνεται διότι τα πρώτα δύο συμπεριφέρονται στη τιμή σαν διεύθυνση και το άλλο σαν απλό νούμερο επειδή το κάνει casting σε integer.

Το μέγεθος του στοιχείου είναι 4 byte και το μέγεθος του πίνακα είναι 10 φορές το στοιχείο άρα 40.

B. Επειδή η κάθε μεταβλητή είναι διαφορετικού τύπου έχουν άλλο μέγεθος άρα και οι διευθύνσεις τους έχουν διαφορετικές αποστάσεις μεταξύ τους. Επίσης αποθηκεύονται διαδοχικά από κάτω προς τα πάνω.

C. Το πρώτο struct δεσμεύει 12 byte και το δεύτερο 8 λόγω της σειράς των στοιχείων τους. Αυτό γίνεται γιατί δεν μπορούν να αποθηκευτούν στην ίδια τετράδα στοιχεία διαφορετικού τύπου. Στο πρώτο struct δεσμεύει 4 byte για char, 4 για int και άλλα 4 για char. Το δεύτερο δεσμεύει 4 byte για char και 4 για int. Έτσι βλέπουμε την καλύτερη δήλωση μεταβλητών σε struct και ότι δεν χρειάζεται η δημιουργία αντικειμένων για να δούμε το μέγεθος τους.

D. Η malloc δεσμεύει περισσότερο χώρο από αυτόν που περιμέναμε. Ο λόγος είναι πως η malloc χρειάζεται να ξέρει πόσο είναι το μέγεθος της δεσμευμένης μνήμης και πρέπει να το αποθηκεύσει σε μερικά bytes.

E. Η διεύθυνση της main είναι πιο κοντά στη διεύθυνση της καθολικής μεταβλητής σε σχέση με τις άλλες. Οι διευθύνσεις της τοπικής και της δεσμευμένης από την malloc μεταβλητής είναι πολύ κοντά σε σχέση με τις άλλες.

0x4010e0 global μεταβλητή	Stack
0x4071a0 main	
0xffffcbcc τοπική μεταβλητή	
0xffffcc08 malloc μεταβλητή	Heap

Συμπεράσματα

Μάθαμε το τρόπο που συμπεριφέρεται η μνήμη και που αποθηκεύονται οι διάφορες μεταβλητές.

Παράρτημα - Κώδικας

A

```
int step(void){

   static int m = -1;
```

```
   int n = -1;
   printf("m is %d, n is %d. ", ++m, ++n);
   return m;}
void main(void){
   int A[10], i; /* A = array of 10 ints, i = scalar int variable */
   int * p; /* p is a scalar variable that points to an int */
   for (i = 0; i < 10; i++){
      A[i] = step();
      printf("Element A[%d] = %d is stored in address: 0x%x\n", i, A[i], &A[i]);
   }
   printf("decimal: %d %d %d %d\n",A,A+1,(int)A+1,&A[1]);
   printf("hexadecimal:%#x %#x %#x %#x\n",A,A+1,(int)A+1,&A[1]);
   printf("%d %d",sizeof(A),sizeof(A[0]));
}
```

В

```
int main(){
   int y = 1 ;
    float z = 1;
   char x = '1';
   printf("%d %d %d",&y,&z,&x);
}
```

Г

```
struct a{
   char X;
   int C;
   char Y;};

struct b{
   char X;
```

```
    char Y;
    int C;};

int main(){
    struct a a;
    struct b b;
    printf("%d %d",sizeof(a),sizeof(b));
}
```

Δ

```
int main(){
    char *x ;
    char *y ;
    char *z ;
    char *w ;

    x =  malloc(1);
    y =  malloc(10);
    z =  malloc(16);
    w =  malloc(32);
    printf("%d %d %d %d",&x,&y,&z,&w);
}
```

E

```
int global;

int main(){
    int *y;
    y = malloc(1);
    printf("%#x %#x %#x ",main,&global,&y);
    synarthsh();
```

```
    return 0;
}


void synarthsh(){
    int x;
    printf("%#x",&x);
}
```