Pointers

amitabha mukerjee

march 22, 2012

Pointer arithmetic

- Pointer addition:

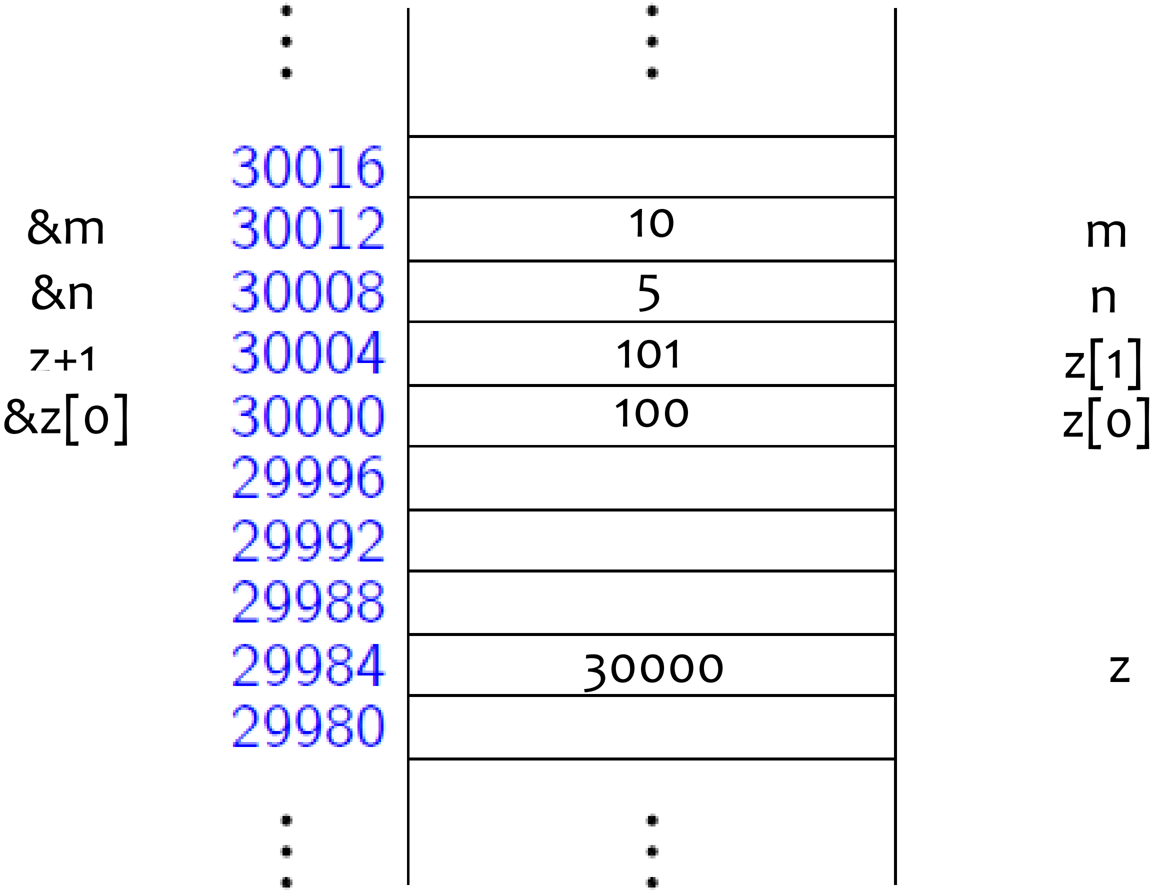
```
int p[2];
```

&p[0] is the same as p

p++ increments p by sizeof(*p);

p1 – p2 = number of steps of sizeof(*p)

Pointer arithmetic



	⋮	⋮	
	30016		
&m	30012	10	m
&n	30008	5	n
z+1	30004	101	z[1]
&z[0]	30000	100	z[0]
	29996		
	29992		
	29988		
	29984	30000	z
	29980		
	⋮	⋮	

```
int main()
{
    int m = 10;
    int n = 5;
    int z[2];
    int i;

    for (i=0; i<2; i++)
        z[i] = 100 +i ;
}
```

based on esc101 lectures 2010 manindra agarwal

Pointer arithmetic

- Pointer addition:

  int p[2];

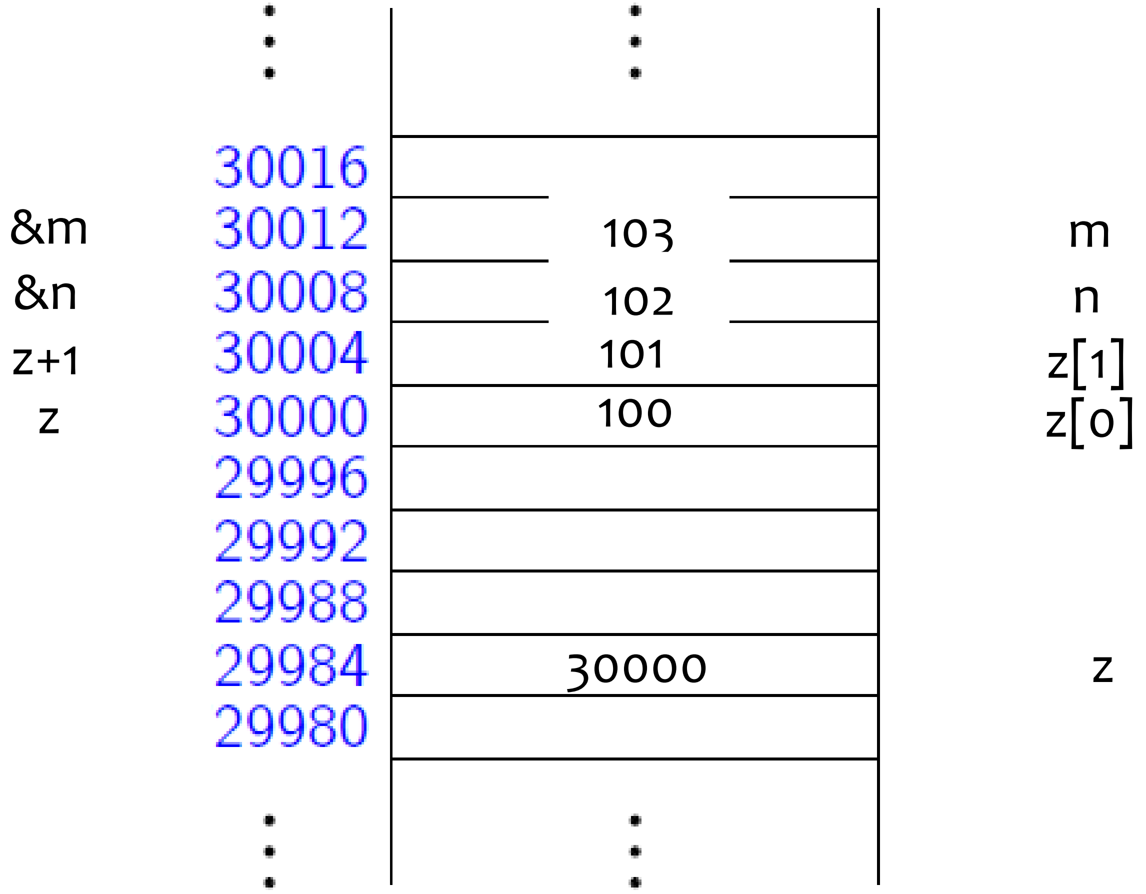
  &p[0] is the same as p

  p++ increments p by sizeof(*p);

  p1 – p2 = number of steps of sizeof(*p)

- But one must be careful!!
  - Can overshoot array bounds without warning

Pointer arithmetic



```
int main()
{
    int m = 10;
    int n = 5;
    int z[2];
    int i;

    for (i=0; i<4; i++)
        z[i] = 100 +i ;
}
```

based on esc101 lectures 2010 manindra agarwal

Pointer arithmetic

- Pointer addition:

  int p[2];

  &p[0] is the same as p

  p++ increments p by sizeof(*p);

  p1 – p2 = number of steps of sizeof(*p)

- But one must be careful!!
  - Can overshoot array bounds without warning
  - What is the value of p Itself?

    Try printing “p” (file: `pointer_overflow.c`):

    [never use this value (an address) – can change from run to run]

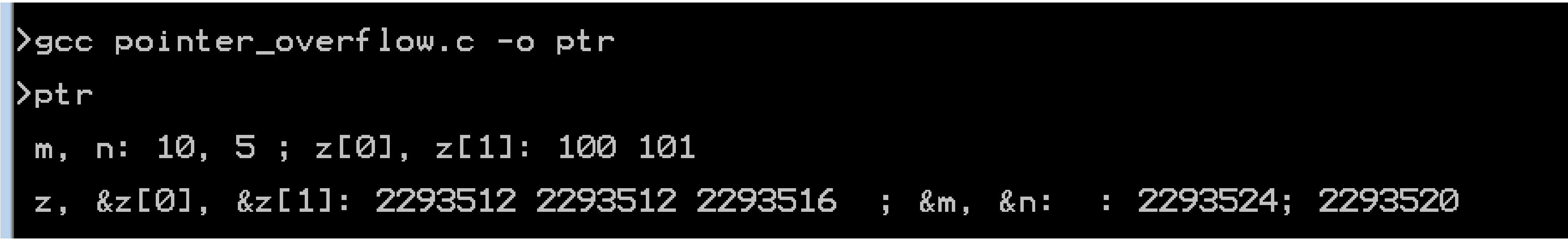
pointer_overflow.c

```
#include <stdio.h>

int main()
{
    int m = 10;
    int n = 5;
    int z[2];
    int i;

    int SIZE=4;     /* overflows; compare with SIZE=2 */
    for (i=0;i<SIZE;i++)
        z[i] = 100 +i;

    printf("\n m, n: %d, %d ; z[0], z[1]: %d %d \n",  m, n, z[0], z[1]);

    printf("\n z, &z[0], &z[1]: %d %d %d  %; &m, &n:  : %d; %d \n", z, &z[0], &z[1], &m, &n);
}
```

pointer_overflow.c

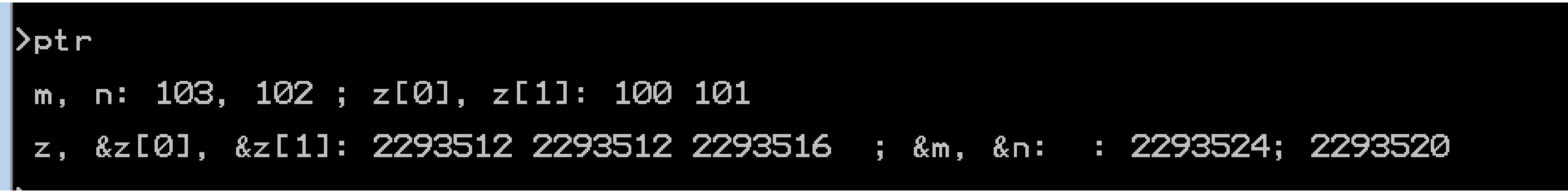
With

int SIZE=2;



```
>gcc pointer_overflow.c -o ptr
>ptr
 m, n: 10, 5 ; z[0], z[1]: 100 101
 z, &z[0], &z[1]: 2293512 2293512 2293516  ; &m, &n:  : 2293524; 2293520
```

int SIZE=4;



```
>ptr
 m, n: 103, 102 ; z[0], z[1]: 100 101
 z, &z[0], &z[1]: 2293512 2293512 2293516  ; &m, &n:  : 2293524; 2293520
```

Multi-dimensional arrays

- All arrays in C are implemented as pointers
- In a multi-dimensional array e.g.

  float average[12][3]

  average points to a bank of 12 pointers, each pointing to an array of 3 floats.
- Each row average[i] is an 1-D array of 3 elements
- **average or *(*average)) is the [0][0] element
- average+1 : what does this represent?
- *(*(average+i)+j) is equivalent to average[i][j]
- float marks[12][30][3]; what is ***marks?

array2d.c

```
#include <stdio.h>

int main()
{
    int i,j, SIZE1=2, SIZE2=3;
    int avg[SIZE1][SIZE2];
    int *x;

    for (i=0;i<SIZE1;i++)
       for (j=0;j<SIZE2;j++)
          *(*(avg+i)+j) = 50 +i*SIZE2 +j;

    printf(" \n addr avg,  *avg, **avg= %d %d %d; \n", avg, *avg, **avg );

    for (i=0;i<SIZE1;i++)
      {
         printf(" \n addr &avg[%d]= %d; values: ", i, &avg[i]);
         x = avg[i];
         for (j=0;j<SIZE2;j++)
              printf("[%d][%d]: %d  ", i, j, *x++);
      }
}
```