NWEN 241
Arrays and Pointers III

Qiang Fu

School of Engineering and Computer Science
Victoria University of Wellington

This Lecture

- Various topics about arrays and pointers
  - Pointer arithmetic
  - Arrays of pointers
  - Pointers to arrays
  - Pointers to pointers
  - Multi-dimensional arrays and pointers

Pointer Arithmetic

- If ptri is a pointer to int i, then (ptri+1) is the address for storing/accessing the next int
  - We can do: ptri += n, ptri++, --ptri, etc.

Pointer Arithmetic

- An example (reverse printing)

```
#define SIZE 10
...

int main(void)
{ ...
  for (i=0; i<SIZE; i++)
    x[i] = i;
  rprint(x);         /* call for reverse printing */
  return 0;
}

void rprint(int *a)        /* void rprint(int a[])
  */
{ int *p=a+SIZE-1;         /* p=&a[9] */
  for (p; p>=a; p--)       /* decrement p */
    printf("x[%d]=%d, &x[%d]=%x\n", \
      (p-a), *p, (p-a), p);
}
```

Pointer Arithmetic

- As mentioned earlier, an array name is a pointer

```
int i[10];
/* i is a pointer to int */
/* void rprint(int a[]) */
/* void rprint(int *a) */
```

- And, we can do pointer arithmetic

```
int *p = i; p++; --p;
```

- Can we do

```
i++; --i;
```


Pointer Arithmetic

- As mentioned earlier, an array name is a pointer

```
int i[10];
/* i is a pointer to int */
/* void rprint(int a[]) */
/* void rprint(int *a) */
```

- And, we can do pointer arithmetic

```
int *p = i; p++; --p;
```

- Can we do

```
i++; --i; /* i is a pointer value */
```


What's this?

```
int *i[10]; /* a pointer to an array? */
            /* an array of pointers? */
```


Arrays of Pointers

- Arrays of pointers
  - The elements in an array are pointers

```
int *i[10]; /* the 10 elements in i are */
            /* pointers to int */
```

  - i[0] is …
  - *i[0] is …

Arrays of Pointers

- Arrays of pointers
  - The elements in an array are pointers

```
int *i[10]; /* the 10 elements in i are */
            /* pointers to int */
```

  - i[0] is an int pointer
  - *i[0] is the integer that i[0] points to

What's this?

```
int (*i)[10];     /* similar to *i[10]? */
```


Pointers to Arrays

- Pointers to arrays

```
int (*i)[10];     /* a pointer to a */
                  /* 10-element array */
int a[10];
i = …;            /* &a or a? */
```


Pointers to Arrays

- Pointers to arrays

```
int (*i)[10];     /* a pointer to a */
                  /* 10-element array */
int a[10];
i = &a;           /* &a or a? */
```

Pointers to Arrays

- Pointers to arrays

```
int (*i)[10];     /* a pointer to a */
                  /* 10-element array */
int a[10];
i = &a;           /* a is an array */
```


Pointers to Pointers

- A pointer may point to another pointer

```
int q;
int *pq = &q;     /* pq points to q */
int **ppq = &pq;  /* ppq points to pq, */
                  /* which points to q */
```


Pointers to Pointers

- A pointer may point to another pointer

```
int q;
int *pq = &q;     /* pq points to q */
int **ppq = &pq;
```


Pointers to Pointers

- A pointer may point to another pointer

```
int *p[10];
/* What is p? */

/* When you pass p to a function, */
/* what are you passing? */
```

Pointers to Pointers

- A pointer may point to another pointer

```
int *p[10];
/* What is p? */

/* When you pass p to a function, */
/* what are you passing? */

void func(int *p[]);

/*or */

void func(int **p);
```


Multi-dimensional Arrays and Pointers

- One-dimensional arrays

```
int i[2] = {2, 1};
```

- A one-dimensional array of one-dimensional arrays is a two-dimensional array

```
int i[2][3]= {{4,1,2}, {9,8,5}};
```

- A one-dimensional array of two-dimensional arrays is a three-dimensional array

```
int i[3][2][4] =  {{{5,4,2,3}, {9,5,1,7}},
                   {{3,7,5,4}, {6,7,6,8}},
                   {{5,4,7,1}, {5,5,4,0}}
                  };
```


Multi-dimensional Arrays and Pointers

- Use *pointers to arrays* to express two-dimensional arrays

```
char week[7][10] = {“Mon”, “Tue”, ...};
char (*ptrw)[10];
ptrw = week; /* what does this mean? */
ptrw++;      /* what does this mean? */
             /* week++;? */
```


Multi-dimensional Arrays and Pointers

- Use *pointers to arrays* to express two-dimensional arrays

```
char week[7][10] = {“Mon”, “Tue”, ...};
char (*ptrw)[10];
ptrw = week; /* points to the first row */
ptrw++;      /* points to the second row */
             /* week++;? Wrong! */
```

  – the row length is fixed 10 bytes

```
sizeof(week);
```

Multi-dimensional Arrays and Pointers

- Use *pointers to arrays* to express two-dimensional arrays

```
char week[7][10] = {“Mon”, “Tue”, ...};
char (*ptrw)[10];
ptrw = week; /* points to the first row */
ptrw++;      /* points to the second row */
             /* week++;? Wrong! */
```

– the row length is fixed 10 bytes

```
sizeof(week);     /* 7*10 = 70 */
                  /* week is not a ptr */
                  /* week is an array */
```


Multi-dimensional Arrays and Pointers

- Use *pointers to arrays* to express two-dimensional arrays

```
char week[7][10] = {“Mon”, “Tue”, ...};
char (*ptrw)[10];
ptrw = week; /* points to the first row */
ptrw++;      /* points to the second row */

/* ptrw – week = ? */

/* (int)ptrw – (int)week = ? */
```


Multi-dimensional Arrays and Pointers

- Use *pointers to arrays* to express two-dimensional arrays

```
char week[7][10] = {“Mon”, “Tue”, ...};
char (*ptrw)[10];
ptrw = week; /* points to the first row */
ptrw++;      /* points to the second row */

/* ptrw – week = 1 */

/* (int)ptrw – (int)week = 10 */
```


Multi-dimensional Arrays and Pointers

- Use *arrays of pointers* to express two-dimensional arrays

```
char *a_week[7] = {“Mon”, “Tue”, ...};
```

Multi-dimensional Arrays and Pointers

- Use *arrays of pointers* to express two-dimensional arrays

```
char *a_week[7] = {“Mon”, “Tue”, ...};
```

  – the row length depends on the actual needs (e.g., 4 bytes for Mon\0)

```
sizeof(week); /* 7*10 = 70 */
```

  – the row length is fixed 10 bytes

Next Lecture

- More on arrays, pointers and functions