Computer Programming 1 Lab

2022 / 11 / 03

Chen, Yi-Hsuan

Outline

- UNIX Commands
- Conditional statements
- Loop
- Recursive
- Array
- Pointer
- Bitwise Operators
- Tips
- Bonus

Basic UNIX Command

`~` => your home directory

`~x` => x's home directory

`.` => current directory

`..` => parent directory

Absolute path: Start with "/"

- /usr/share/bin
- /home1/student/stud110/s110xx

Relative path: Path relate to current directory.

- If current dir is /usr
  test/bin => /usr/test/bin
  li/public => /usr/li/public

Basic UNIX Command

- `ls` list files in current directory.

```
ls
ls -l  # list files details in current directory.
ls -a  # list all files (include hidden files).
ls -al # Both of listing all files with details.
```

Basic UNIX Command

How to create/delete/copy files or directories?

```
touch a.txt
# Create an empty file named "a.txt" in current directory.
mkdir test
# Create a directory named "test" in current directory.
cp fileX dirY/dirZ
# Copy fileX from current directory to ./dirY/dirZ
cp fileX dirY/fileZ
# Copy fileX from current directory to dirY and rename to fileZ.
cp -r dirX dirY
# Copy dirX from current directory to dirY.
# If dirY doesn't exist, dirY is a copy of dirX.
# If dirY is a directory then there will be a copy of dirX under dirY.
```

Basic UNIX Command

How to create/delete/copy files or directories?

```
mv fileA dirB
# Move fileA to dirB.
mv dirA dirB
# If dirB exist, then move dirA under dirB.
# If dirB does not exist, dirA is rename to dirB.
rm fileA
# Remove file fileA (Only for file).
rm -r dirA
# Remove directory dirA and all its contents.
```

Basic UNIX Command

Use `mkdir` to create a directory

Use `mv` to move a directory/file or rename a directory/file

Use `cp` to copy a file and `cp -r` to copy a directory

Use `rm` for removing file and `rm -r` for removing directory

Basic UNIX Command

Pipe

usage: [command A] (`&&` , `|` , `||`) [command B]

- `A &` run ***A*** background
- `A && B` run ***B*** when ***A*** excute success
- `A | B` use ***A***'s output as ***B***'s input
- `A || B` run ***B*** when ***A*** excute fail

Basic UNIX Command

Redirection

usage: [command A] (`<` , `<<` , `>` , `>>`) [filename X]

- `A < X` let ***X***'s content to be ***A***'s stdin.
- `A << X` read next n lines(like edit a file) until ***X*** appear.
- `A > X` let ***A***'s output to be a file named ***X***(rewrite).
- `A >> X` let ***A***'s output append in a file named ***X***(append at last line).

Conditional statements

Boolean data type

- Two states (true or false) / (1 or 0)
- Logical operators - AND(`&&`), OR(`||`), NOT(`!`)

Conditional statements

if / else statement

examples:

```
if((1 + 1 == 2) && (1 + 1 == 3)){ // return false
    // This part will NOT be executed.
}
if((1 + 1 == 2) || (1 + 1 == 3)){ // return true
    // This part will be executed.
}
if(!(1 + 1 == 3)){ // return true
    // This part will be executed.
}
```

Conditional statements

if / else statement

what "else"

- Can only be used with `if()`.
- Executed when the previous `if()` does not executed.

Conditional statements

if / else statement

examples:

```
if(Letter == 'A'){
    // Do something
}
else if(Letter == 'B'){
    // Do something
}
else if(Letter == 'C'){
    // Do something
}
else{
    // Do something
}
```

Conditional statements

"switch" on

- switch between cases
- `break` each cases
- Use `default` as the last `else`

Conditional statements

switch statement

examples:

```
switch(Letter){
    case 'A':
        // Do something
        break;
    case 'B':
        // Do something
        break;
    case 'C':
        // Do something
        break;
    default:
        // Do something
        break;
}
```

Conditional statements

"switch" Tips:

- Don't forget to break.

Loop

For loop

- Usage: `for(init; condition; increment){}`
- Init part will be executed before for loop start.
- Condition part will be executed before each looped. Only when return value is true will the next loop be triggered.
- Increment will be executed after each loop.

Loop

For loop

examples:

```
for(int a = 0; a < 5; a++){
    printf("%d\n", a);
}
```

results:

```
0
1
2
3
4
```

Loop

"For" Pro Tips:

1. Declear an int and start with 0, set condition as `index < N;` and increment as `index++`. This for loop will run N times with index = 0, 1, 2, 3.....N-2, N-1.
2. If you get a segmentation fault during runtime, it may because your for loop messed up. For example, `for(int index = N-1; index >= 0; index++)`.
3. You may declare multiple variables in init part by using `int a = 0, b = 0, ...;`. Please note that they should be the same data type.

Loop

While Loop

- Usage: `while(condition){statement(s)}`.
- While (condition == true), do statement(s), then do the whole loop again.

In conclution, this is how it works...

if(condition == true) -> execute { } ->
if(condition == true) -> execute { } ->
...
if(condition == false) -> leave while()

Loop

While Loop

example:

```
int total = 100;
while(total != 0){
    printf("%d ", total);
    total /= 2;
}
printf("\n");
```

results:

```
100 50 25 12 6 3 1
```

Loop

"While" Pro Tips:

1. If your runtime is stucked, it is very possible that you have an infinite while loop. For example, `while(a > 1){printf("%d ", a);}`. The value of `a` won't be changed in the loop, so if you enters this while loop, it's gonna run FOREVER.

Array

- Sort Array
- Search in Array

Array

- Sort Array (Bubble sort)

```
int n = 5;
int arr[5] = {2, 3, 5, 1, 4};
for(int i=0; i<n-1; ++i)
    for(int j=0; j<n-i-1; ++j)
        if(arr[j] > arr[j+1])
            arr[j]^=arr[j+1], arr[j+1]^=arr[j], arr[j]^=arr[j+1];
```

Array

- Search in Array (Linear search)

```
int n = 5;
int arr[5] = {2, 3, 5, 1, 4};
int target = 4;
for(int i=0; i<n; ++i){
    if(arr[i] == target){
        printf("index = %d\n", i);
        break;
    }
}
```

Array

- Search in Array (Binary search)

```
int BS(const int arr[], int tar, int n){
    int mid;
    while(l <= r){
        mid = (l + r) / 2;
        if(tar == arr[mid]) return mid;
        else if(tar < arr[mid]) r = mid - 1;
        else l = mid + 1;
    }
    return -1;
}
```

```
int n = 8;
int arr[8] = {1, 2, 3, 4, 5, 6, 7, 8};
int tar = 7;
printf("%d\n", BS(arr, tar, n)); // 6
```

STEP	---1---	---2---	---3---	---4---	---5---	---6---	---7---	---8---
(1)	L							H
				(M)				
(2)					L			H
						(M)		
(3)							L	H
							(M)	

Pointer

Pointer

```
int main(){
    int a = 1;
    int b = 2;
    int c = 3;
}
```

Memory Address	Value	Variable
0X0012FF70	1	a
0X0012FF74	2	b
0X0012FF78	3	c

Pointer

- `*`
  i. Pointer (指標)
  Declare that the type of the variable is a pointer. -> It stores **address.**
  ii. Dereference operator (取值運算子)
  Apart from variable declaration, we use `*` to get the value which is stored in the variable's address.
- `&`
  Address-of operator (取址運算子) -> Get the variable's **memory address.**

Pointer

```c
int main(){
    int a = 1;
    int *ptr = &a; // Declare a int pointer named "ptr" and it points to a's address
}
```

Memory Address	Value	Variable
0X0012FF70	1	a
0X0012FF74	0X0012FF70	ptr

- **Pointer variable**: a variable that stores pointer
- **Pointer**: point to a variable's address

Bitwise Operators

`&`, `|`, `^`, `~`, `<<`, `>>`

Bitwise Operators

A = 10 (1010), B = 3 (0011)

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 2 (0010)
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) = 11 (1011)
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 9 (1001)

Bitwise Operators

Operator	Description	Example
~	Binary One's Complement Operator is unary and has the effect of 'flipping' bits.	~A = -11
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 40
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 = 2

Tips

- `a++` vs. `++a`

Bonus - RMQ

It's an optional excerise, a bonus.

Any Questions?