
Structure organization

Read, parse and execute

Group 13

Yong Li

Yulin Gao

Yanhui Meng

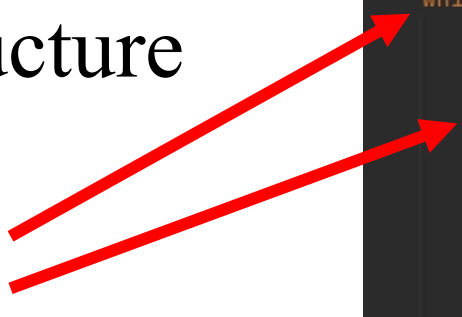
Yiming Ju

A solid orange horizontal bar at the bottom of the slide.

Whole Structure

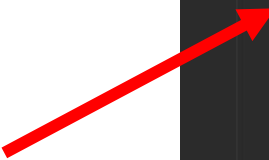
- Read command
- Parse command
- Execute command

```
void shell(){
    while(1){
        //read the command
        scanf("%s", command);
        while(1){
            //parse the command
            //split the command and save in args[]
        }
        //execute the command
        pid_t pid = fork();
        if (pid < 0){
            //stderr
        }
        else if (pid > 0){
            //parent process
        }
        else if (pid == 0){
            //child process
            if (pipe){
                pid_t pipePid = fork();
                //similar to above part
            }
            else{
                execvp();
            }
        }
    }
}
```



Read command


- Get the whole command



```
void shell(){
    while(1){
        //read the command
        scanf("%[^\n]", command);
        while(1){
            //parse the command
            //split the command and save in args[]
        }
        //execute the command
        pid_t pid = fork();
        if (pid < 0){
            //stderr
        }
        else if (pid > 0){
            //parent process
        }
        else if (pid == 0){
            //child process
            if (pipe){
                pid_t pipePid = fork();
                //similar to above part
            }
            else{
                execvp();
            }
        }
    }
}
```

Parse command

- Detect the syntax
- Split the command
- Save the arguments



```
void shell(){
    while(1){
        //read the command
        scanf("%[^\n]", command);
        while(1){
            //parse the command
            //split the command and save in args[]
        }
        //execute the command
        pid_t pid = fork();
        if (pid < 0){
            //stderr
        }
        else if (pid > 0){
            //parent process
        }
        else if (pid == 0){
            //child process
            if (pipe){
                pid_t pipePid = fork();
                //similar to above part
            }
            else{
                execvp();
            }
        }
    }
}
```

Execute command

- CTRL-D, CTRL-C
- Exit, cd, pwd
- Fork a child process
 - Single commands
 - Fife I/O redirection
- Pipe
 - Loop (eg. `echo 123 | grep 1 | grep 1`)

```
void shell(){
    while(1){
        //read the command
        scanf("%s", command);
        while(1){
            //parse the command
            //split the command and save in args[]
        }
        //execute the command
        pid_t pid = fork();
        if (pid < 0){
            //stderr
        }
        else if (pid > 0){
            //parent process
        }
        else if (pid == 0){
            //child process
            if (pipe){
                pid_t pipePid = fork();
                //similar to above part
            }
            else{
                execvp();
            }
        }
    }
}
```

Extension

- Historical command
- wait()
- Background &

```
void shell(){
    while(1){
        //read the command
        scanf("%s", command);
        while(1){
            //parse the command
            //split the command and save in args[]
        }
        //execute the command
        pid_t pid = fork();
        if (pid < 0){
            //stderr
        }
        else if (pid > 0){
            //parent process
        }
        else if (pid == 0){
            //child process
            if (pipe){
                pid_t pipePid = fork();
                //similar to above part
            }
            else{
                execvp();
            }
        }
    }
}
```