

OS Project 1

Tutorial 3

蔡子诺

zinuocai@gmail.com

目录

- I/O 重定向
- 批处理
- 前台/后台执行
- 代码解读
- 测试

I/O 重定向

```
programname arg1 arg2 < inputfile > outputfile
```

或者以下指令：

```
programname arg1 arg2 < inputfile >> outputfile
```

问题1：如何进行文件的读写？

问题2：如何使用文件流替换标准输入输出流？

- 其中，`programname` 是可执行指令，`arg*` 是指令的参数。该指令从 `inputfile` 中获取用户输入，而不是标准输入 `stdin`；指令执行的结果会输出到 `outputfile`，而不是标准输出 `stdout`。
- 输出重定向会和 `dir` `environ` `echo` `help` 含有输出的内部指令共同使用，输出结果会重定向到用户指定的文件中。例如 `help > outputfile` 的执行结果应该输出到用户指定的文件 `outputfile`。
- 当使用输出重定向时 问题3：如何改变“写”的模式？
 - 如果表示重定向的字符串是 `>`，
 - 如果 `outputfile` 不存在，则新建文件。
 - 如果 `outputfile` 存在，那么输出会覆盖原文件。
 - 如果表示重定向的字符串是 `>>`，
 - 如果 `outputfile` 不存在，则新建文件。
 - 如果 `outputfile` 存在，那么输出添加到原文件后面。

I/O 重定向

- 常用函数介绍

- `int feof(FILE *__stream)`
- `char *fgets(char *__restrict __s, int __n, FILE *__restrict __stream)`
- `int access(const char *__name, int __type)`
- `FILE *fopen(const char *__restrict __filename, const char *__restrict __modes)`
- `int fprintf(FILE *__restrict __stream, const char *__restrict __format, ...)`
- `FILE *freopen(const char *__restrict __filename, const char *__restrict __modes, FILE *__restrict __stream)`

提示：子进程在调用 `execvp` 执行指令前，需要使用 `freopen` 切换重定向的输入输出流

批处理


如果Shell工具在使用时带有参数，那么它可以从参数指定的文件中读取指令，并依次执行。例如，当我们这样使用Shell工具时：

```
myshell batchfile
```

提示：

1. 指令的输入流从stdin变成了文件流；
2. 注意理解与I/O重定向的区别。

那么，我们会依次读取 `batchfile` 文件的每一行并执行。当读取到文件的最后一行时，Shell会退出。



前台/后台执行

- 当一条指令后面有& 符号时，Shell不需要等待该指令执行结束才能返回。
- 提示：修改waitpid的options选项设置前台/后台执行。
- `pid_t waitpid(pid_t pid, int *status, int options);`

WNOHANG

return immediately if no child has exited.

WUNTRACED

also return if a child has stopped (but not traced via **ptrace**(2)). Status for *traced* children which have stopped is provided even if this option is not specified.

代码解读

- 1个结构体

```
struct shellstatus_st
{
    int foreground; // foreground execution flag
    char *infile;   // input redirection flag & file
    char *outfile;  // output redirection flag & file
    char *outmode;  // output redirection mode
    char *shellpath; // full pathname of shell
};
typedef struct shellstatus_st shellstatus;
```

- 2个重要函数

- `void check4redirection(char **, shellstatus *)`; // check command line for i/o redirection
- `void execute(char **, shellstatus)`; // execute command from arg array

测试

谢谢！

蔡子诺

zinuocai@gmail.com