Linux 搜索关键词

好的，针对你提供的题目，下面是每个问题的详细解题思路和代码实现。

1. C语言设计命令行交互菜单

设计思路：

- 使用 `switch` 语句或 `if-else` 语句来实现菜单选择。
- 每个菜单选项执行一个对应的系统命令，通过 `popen()` 函数调用系统命令并获取结果。
- 将用户的输入以及显示的信息写入 `log.txt` 文件中。
- 用 `makefile` 来管理项目，确保各个 `.c` 和 `.h` 文件的依赖关系。
- 使用 `gcc` 编译，使用 `gdb` 调试。

代码实现：

c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void log_to_file(const char *log_msg) {
    FILE *log_file = fopen("log.txt", "a");
    if (log_file != NULL) {
        fprintf(log_file, "%s\n", log_msg);
        fclose(log_file);
    } else {
        perror("Error opening log file");
    }
}

void show_cpu_info() {
    printf("CPU Information:\n");
    log_to_file("Option A: View CPU hardware info");
    system("lscpu");
}

void show_process_info() {
    printf("Process Information:\n");
    log_to_file("Option B: View process information");
    system("ps aux");
}

void show_memory_info() {
    printf("Memory Information:\n");
    log_to_file("Option C: View memory hardware and usage info");
    system("free -h");
}

void show_storage_info() {
    printf("Storage Information:\n");
    log_to_file("Option D: View storage hardware and usage info");
    system("df -h");
}

int main() {
    char choice;
    while (1) {
        printf("-----------------\n");
```

```
            printf("C\t\t交互菜单\n");
            printf("-----------------\n");
            printf("A: 查看CPU硬件信息\n");
            printf("B: 查看进程运行信息\n");
            printf("C: 查看内存硬件信息、内存使用信息\n");
            printf("D: 查看存储设备硬件信息、存储设备使用信息\n");
            printf("E: 退出\n");
            printf("-----------------\n");

            printf("请输入选项: ");
            scanf(" %c", &choice);

            switch (choice) {
                case 'A':
                    show_cpu_info();
                    break;
                case 'B':
                    show_process_info();
                    break;
                case 'C':
                    show_memory_info();
                    break;
                case 'D':
                    show_storage_info();
                    break;
                case 'E':
                    printf("退出程序\n");
                    log_to_file("Option E: Exit");
                    exit(0);
                default:
                    printf("无效选项，请重新输入。\n");
            }
        }

        return 0;
    }
```

Makefile 示例：

```
makefile


  CC = gcc
  CFLAGS = -Wall -g

  all: menu_program

  menu_program: menu_program.o
      $(CC) $(CFLAGS) -o menu_program menu_program.o

  menu_program.o: menu_program.c
      $(CC) $(CFLAGS) -c menu_program.c

  clean:
      rm -f *.o menu_program
```

2. Shell脚本程序

设计思路：

- 使用 `**date**` 命令获取系统日期，使用 `**ps**`,`**who**`,`**pwd**`,`**hostname**`,`**ifconfig**` 等命令获取系统信息。
- 使用 `**select**` 语句创建交互式菜单，用户输入对应选项执行操作。
- 使用 `**echo**` 命令输出结果。

Shell脚本实现：

```
bash
```

```
#!/bin/bash

# 显示系统日期
DATE=$(date +"%Y/%m/%d")
echo "Linux    $DATE"

while true; do
    echo "------------------"
    echo "A: 查看进程信息"
    echo "B: 查看用户信息"
    echo "C: 显示当前目录信息"
    echo "D: 显示计算机名、Linux内核版本、当前IP地址"
    echo "E: 退出"
    echo "------------------"
    read -p "请输入选项: " option

    case $option in
        A)
            echo "进程信息："
            ps aux
            ;;
        B)
            echo "用户信息："
            who
            ;;
        C)
            echo "当前目录信息："
            pwd
            ;;
        D)
            echo "计算机名、Linux内核版本、当前IP地址："
            hostname
            uname -r
            ip a | grep inet
            ;;
        E)
            echo "退出程序"
            break
            ;;
        *)
            echo "无效选项，请重新输入。"
            ;;
    esac
done
```

3. Vim命令

设计思路：

- 用 `vi` 打开存在错误的 `.c` 文件，找出所有错误。
- 修改错误，包括变量拼写错误、不正确的分号、括号位置不正确等。
- 使用 `=` 命令调整代码缩进。

可能的错误类型：

- 变量拼写错误：可能是变量名错拼，或者未定义的变量。
- 分号错误：漏掉了语句末尾的分号。

- 括号匹配错误：函数调用的括号未正确配对。

Vim修复步骤：

1. 使用 `vi m1.c` 打开文件。
2. 查找并修复拼写错误。使用 `:set spell` 可以启用拼写检查。
3. 使用 `=` 自动调整缩进，或者手动调整。
4. 确保每个语句末尾都有分号。
5. 使用 `:wq` 保存并退出。

报告内容

报告结构

1. **文件依赖关系图**：可以使用工具如 `dot` 或手动绘制依赖关系图，展示 `.c` 和 `.h` 文件之间的关系。
2. **设计思路**：描述实现思路，包括如何设计菜单、如何调用系统命令等。
3. **编译和执行结果**：
   - 通过 `make` 编译，执行 `./menu_program` 查看程序的运行效果。
   - Shell脚本运行结果。
   - Vim操作结果，显示源代码修改前后的差异。
4. **调试过程和视频**：录制调试过程，展示代码修改、调试的关键步骤。

如果你有其他具体问题或需要进一步的帮助，请告诉我！