# 1.1 项目框架设计与实现_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 1.1 项目框架设计与实现涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

全部开发者教程

## 物联网 / 嵌入式工程师

第 24 周 stm32 芯片 – 智能硬件项目实战与企业笔试

未命名节

第 25 周 大厂必备 – linux 内核与文件系统移植

未命名节

第 26 周 嵌入式开发 – 系统移植 – bootloader、yocto

未命名节

第 27 周 嵌入式底层核心技能 – Linux 设备驱动初级

未命名节

第 28 周 嵌入式底层核心技能 – Linux 设备驱动中级

未命名节

第 29 周 嵌入式底层核心技能 – Linux 设备驱动高级 1

未命名节

第 30 周 嵌入式底层核心技能 – Linux 设备驱动高级 2

未命名节

第 31 周 嵌入式人工智能必备 – Python

未命名节

第 32 周 智能家居项目实战之客户端功能开发

未命名节

第 33 周 智能家居项目实战之网关端功能开发

未命名节

第 34 周 智能家居项目实战之设备端功能开发

未命名节

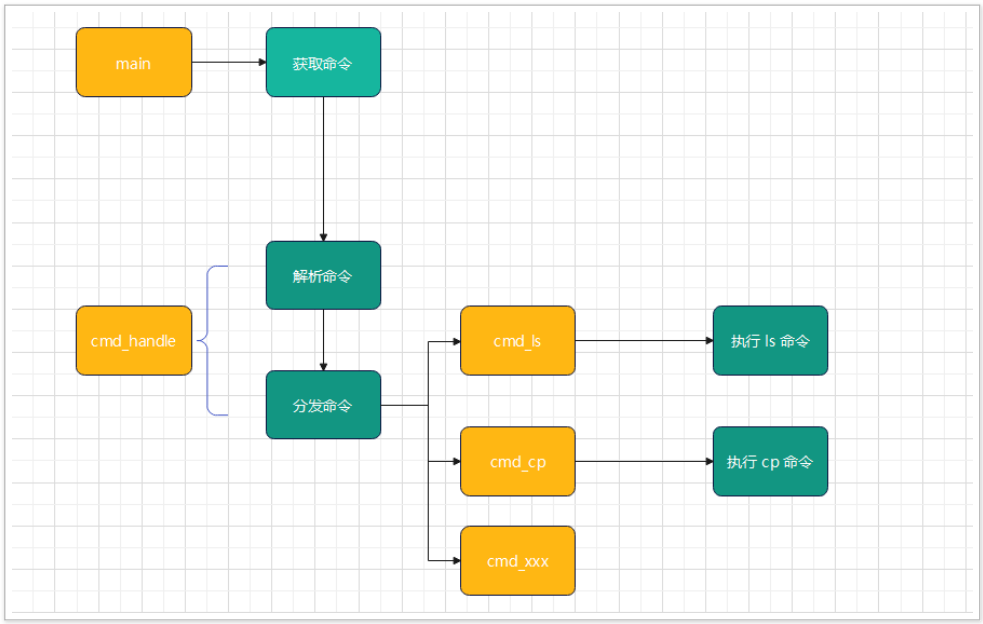
第 35 周 物联网 / 嵌入式项目答辩和就业指导

未命名节

第 36 周 独立开发阶段 – 三大热门领域项目

未命名节

- 实现一个基础的 shell 程序，主要完成两个命令的功能 cp 和 ls
    - cp 命令主要实现:
        - 文件复制
        - 目录复制
    - ls 命令主要实现:
        - ls –l 命令的功能
- 在框架设计上，采用模块化设计思想，并具备一定的可扩展性, 具体框架如下:




- cmd_handle 模块: 用于解析命令相关信息，并进行命令的分发执行
- cmd_ls 模块：用于执行 ls 命令
- cmd_cp 模块：用于执行 cp 命令
- cmd_xxx 模块：用于扩展

| 模块 | 源文件 |
| --- | --- |
| 命令处理中心模块 | cmd_handle.c cmd_handle.h |
| ls 命令模块 | cmd_cp.c cmd_cp.h |
| cp 命令模块 | cmd_ls.c cmd_ls.h |
| 工程管理 | Makefile |
| 主函数 | main.c |

- step 2：创建 Makefile 用于管理工程
    - ```
      OBJS := main.o cmd_ls.o cmd_cp.o cmd_handle.o
      TARGET := tinyshell

      $(TARGET): $(OBJS)
           @gcc   $^ -o $@
           @echo "Done."


      %.o:%.c
          @gcc -c $< -o $@
      ```
      ```

```
clean:
    rm -rf *.o $(TARGET)
```

- := 表示当前位置所赋的值，而不是整个 Makefile 展开之后的值,= 是整个 Makefile 展开之后的所赋的值
    - = 号示例
        - ```
          x = foo
          y = $(x) bar
          x = xyz
          ```
        - 上述示例中的 y 的值为 xyz bar
    - : = 号示例
        - ```
          x := foo
          y := $(x) bar
          x := xyz
          ```
        - 上述示例中的 y 的值为 foo bar
- 
- $(TARGET) : 表示获取 TARGET 变量的值
- %.o :%.c :
    - % 表示通配符
    - %.o : 用于匹配任意 .o 文件, 如 cmd_handle.o ,cmd_ls.o ,…
    - %.c : 用于匹配任意 .c 文件, 如 cmd_handle.c ,cmd_ls.c ,…
- 

- step 3 : 在 main.c 编写基本的 main 函数

```
int main()
{
    return 0;
}
```

- step 4 : 编译测试
    - 在命令行输入 make 命令进行测试, 显示 Done , 则表示编译通过
- 项目的主循环主要完成的功能:
    - step 1 : 循环获取用户输入 main.c
    - ```
      int main(void)
      {
          char command[SZ_CMD] = {0};


          for(;;){

              printf("TinyShell > ");

              fgets(command,SZ_CMD,stdin);

              command[strlen(command) - 1] = '\0';

              if (strncmp(command,"quit",4) == 0){
                  printf("GoodBye\n");
                  break;
              }

              cmd_execute(command);
          }

          return 0;
      }
      ```
    - step 2 : 调用 cmd_handle 的 cmd_execute 接口执行相应的命令 cmd_handle.h cmd_handle.c
        - cmd_handle.h
        - ```
          #ifndef __CMD_HANDLE_H_
          #define __CMD_HANDLE_H_
          ```

```
#define DEBUG

extern int cmd_execute(char *cmd_str);


#endif
```

- cmd_handle.c
- ```
  int cmd_execute( char *cmd_str)
  {

  #ifdef DEBUG
    printf("[DEBUG] : cmd string : < %s >\n",cmd_str);
  #endif
      return 0;
  }
  ```

- step 3： 编译并执行工程
-

全文完




  ```