1.员工管理系统

1.1服务器（多进程，多线程，IO多路复用epoll）

1. 创建socket

```
socket=socket(AF_INET,SOCK_STREAM,0);

```

填充网络信息结构体

```
struct sockaddr ->通用的结构体兼容ipv4,ipv6,本地通信

struct sockaddr_in saddr;

saddr.sin_family = AF_INET;

saddr.sin_addr.s_addr = inet_addr(argv[1]);

saddr.port = htons(atoi(argv[2]));
```

2. 绑定

```
bind(sockfd,(struct sockaddr *)&saddr,sizeof(saddr));
```

3. 设置服务器为监听状态

```
listen(sockfd,5);
```

4. 注册信号处理函数，回收子进程资源

```
void handle(int signo) //信号处理函数
{
    //当子进程结束的时候，父进程收到SIGCHLD信号，使用非阻塞方式回收子进程的资源
    waitpid(-1,NULL,WNOHANG);
}

if（SIG_ERR==signal(SIGCHLD,handle)）
    PRINT_ERR("signal error")
```

5. 等待客户端连接

```
while(1){
    struct sockaddr_in caddr;
    socklen_t len = sizeof(caddr);
    acceptfd = accept(sockfd,(struct sockaddr *)&caddr,&len);
```

6. 创建多进程

```
    pid = fork();
    if(pid==-1){
        PRINT_ERR("fork error");
    }else if(pid==0){
         //子进程，处理客户端的请求
        child_handle_request();
    }
}
```

7. 在子进程中处理客户端的请求

```
typedef struct{
    char type;//'R','L','C','Q','M',H'
    char info[1024];
}msg_t;
```

```
msg_t msg;
while(1){
    recv(acceptfd,&msg,sizeof(msg),0);
    switch(msg.type){
        case 'R':
            break;
        case 'L':
            break;
        case 'C':
           break:
        case 'M':
            break;
        case 'H':
            break;
        case 'Q':
            //结束子进程
            //exit(0);
            break;
    }
}
```

8. 数据库的操作

参考数据库操作文档

1.2客户端

1. 创建套接字

```
sockfd = socket(AF_INET,SOCK_STREAM,0);
```

2. 填充服务器的信息结构体

```
struct sockaddr ->通用的结构体兼容ipv4,ipv6,本地通信
struct sockaddr_in saddr;
saddr.sin_family = AF_INET;
saddr.sin_addr.s_addr = inet_addr(argv[1]);
saddr.port = htons(atoi(argv[2]));
```

3. 连接服务器

```
connect(sockfd,(struct sockaddr *)&saddr,sizeof(saddr));
```

4. 发送请求数据

```
while(1){
    puts("*********************************");
    puts("******1.注册，2登录，3退出*********");
    puts("*********************************");
    //用户数据选择
    switch(chose){
        case 1:
            输入用户信息=>msg_t msg;
            send(sockfd,&msg,sizeof(msg),0);
        case 2:
            输入用户信息=>msg_t msg;
            send(sockfd,&msg,sizeof(msg),0);
            如果登录成功执行接下来的操作
            client_handle();
        case 3:
            //告诉服务器，客户端退出，服务器对应子进程退出，被父进程回收资源
            send(sockfd,&msg,sizeof(msg),0);
            break;
    }

}
```

5. 登录成功的操作逻辑

```
while(1){
    puts("*********************************");
    puts("******1.查询，2.修改，3历史，4退出**");
    puts("*********************************");
     switch(chose){
            case 1:
            case 2:
            case 3:
            case 4:
        }
}
```