# 第 4 天作业

## 编程操作（必做）

1. 电网 SCADA 通信原型系统

   - 定义电网公司 EMS 和变电站 RTU 侧的通信相关数据表结构
   - 开发 EMS 侧通信程序、开发 RTU 侧通信程序。
   - EMS 主站具备功能：python ems.py

     1. 执行 python ems.py init， 重建数据库
     2. 遍历 ems_rtu_info，通过多线程的方式，同时连接多个 RTU 服务端程序
     3. 接收 RTU 传送过来的变化遥测、变化遥信，更新本地数据库，更新结果返回子站
     4. 检测本地数据库里的遥调和遥控（时间或值）的变化，向 RTU 发送变化后的遥调、遥控，更新结果返回主站。
     5. 以 5 秒为周期，执行总召功能（更新所有遥测和遥信）

   - RTU 子站具备功能：python rtu.py

     1. 执行 python rtu.py n init， 重建第 n 个子站的数据库
     2. 执行 python rtu.py n， 启动第 n 个子站的服务
     3. 执行 python rtu.py init， 重建 5 个子站的数据库
     4. 执行 python rtu.py， 通过多进程的方式，同时启动所有 5 个子站的服务
     5. 针对每一个 RTU 子站，检测本地数据库里遥测和遥信（时间、值或状态）的变化，向 EMS 上传变化后的遥测、变化遥信，更新结果返回子站
     6. 接收 EMS 传送过来的变化遥调、遥控信息，更新本地数据库，更新结果返回主站

**rtu.py，只更新变化的遥测和遥信：**

```python
yc_dic = {}

def check_yc_change(li):
    find_ele = yc_dic.get(li[0], None)
    if find_ele is None \
            or abs(find_ele[2] - li[2]) > 1e-6 \
            or find_ele[3] != li[3] \
            or find_ele[4] != li[4]:
        yc_dic[li[0]] = li
        return True
    return False

def update_yc_data(sqldb, conn):
    results = sqldb.execute(text("select * from rtu_yc_info"))
    data = []
    for result in results:
        li = [result.id, result.name, result.value,
              result.status, result.refresh_time]
        if check_yc_change(li):
            data.append(li)

    if len(data) > 0:
        send_data(conn, {"type": "update_yc"}, data)
```

```python
yx_dic = {}

def check_yx_change(li):
    find_ele = yx_dic.get(li[0], None)
    if find_ele is None or find_ele != li:
        yx_dic[li[0]] = li
        return True
    return False

def update_yx_data(sqldb, conn):
    results = sqldb.execute(text("select * from rtu_yx_info"))
    data = []
    for result in results:
        li = [result.id, result.name, result.value,
              result.status, result.refresh_time]
        if check_yx_change(li):
            data.append(li)

    if len(data) > 0:
        send_data(conn, {"type": "update_yx"}, data)
```

**ems.py，只更新变化的遥控和遥调：**

```python
yk_dic = {}
def check_yk_change(rtu_id, yk):
    find_ele = yk_dic.get((rtu_id, yk[0]), None)
    if find_ele is None or find_ele != yk:
        yk_dic[(rtu_id, yk[0])] = yk
        return True
    return False

def update_yk_data(sqldb, sock, rtu_id):
    print("update_yk_data ", rtu_id)
    results = sqldb.execute(text(f"select * from ems_yk_info "
                                 f"where rtu_id = {rtu_id}"))
    data = []
    for result in results:
        print(result)
        yk = [result.pnt_no, result.name, result.value,
              result.ret_code, result.ctrl_time]
        if check_yk_change(rtu_id, yk):
            data.append(yk)

    if len(data) > 0:
        print("end update_yk_data ", rtu_id)
        send_data(sock, {"type": "update_yk"}, data)
```

```python
yt_dic = {}
def check_yt_change(rtu_id, yt):
    find_ele = yt_dic.get((rtu_id, yt[0]), None)
    if find_ele is None \
            or abs(find_ele[2] - yt[2]) > 1e-5 \
            or find_ele[3] != yt[3] \
            or find_ele[4] != yt[4]:
        yt_dic[(rtu_id, yt[0])] = yt
        return True
    return False

def update_yt_data(sqldb, sock, rtu_id):
    results = sqldb.execute(text(f"select * from ems_yt_info "
                                 f"where rtu_id = {rtu_id}"))
    data = []
    for result in results:
        yt = [result.pnt_no, result.name, result.value,
              result.ret_code, result.ctrl_time]
        if check_yt_change(yt):
            data.append(yt)

    if len(data) > 0:
        send_data(sock, {"type": "update_yt"}, data)
```

**rtu.py 的多进程代码示意：**

```python
if __name__ == "__main__":
    rtu_id = -1
    b_init = False
    if "init" in sys.argv:
        b_init = True
        sys.argv.remove("init")

    if len(sys.argv) > 1:
        rtu_id = int(sys.argv[1])

    if rtu_id > 0:
        rtu_main(rtu_id, b_init)
        exit()

    for rtu in range(1, 6):
        process = Process(target=rtu_main, args=(rtu, b_init))
        process.start()
```

```python
def rtu_main(rtu_id, b_init):
    engine = create_engine(f"sqlite:///db/rtu_{rtu_id}.db")
    if b_init:
        auto_gen_tables(engine, rtu_id)
        return
    rtu_addr = get_ip_addr(engine, rtu_id)
    print("rtu_addr:", rtu_addr)
    sock = socket.socket()
    sock.bind(rtu_addr)
    sock.listen(10)

    while True:
        conn, status = sock.accept()
        thread = Thread(target=rtu_thread_update_data, args=(engine, conn))
        thread.start()
        thread = Thread(target=rtu_thread_recv_data, args=(engine, conn))
        thread.start()
```