# 服务器操作手册

该文档与存放在705电脑上的旧版操作手册（以下简称旧操作文档）配合使用。

该文档适用于2023.7.1这个服务器版本。该版本跟上位机配合使用，故有些地方跟旧操作文档有出入。

**1，硬件及网络环境**

三个节点，ip地址分别为192.168.1.150, 192.168.1.151, 192.168.1.152,其中192.168.1.150是主节点。

各节点中，/home/xianyun/XD\_YLSim/code/usr\_lib/ 目录下存放用户编译出的仿真用的动态链接库。

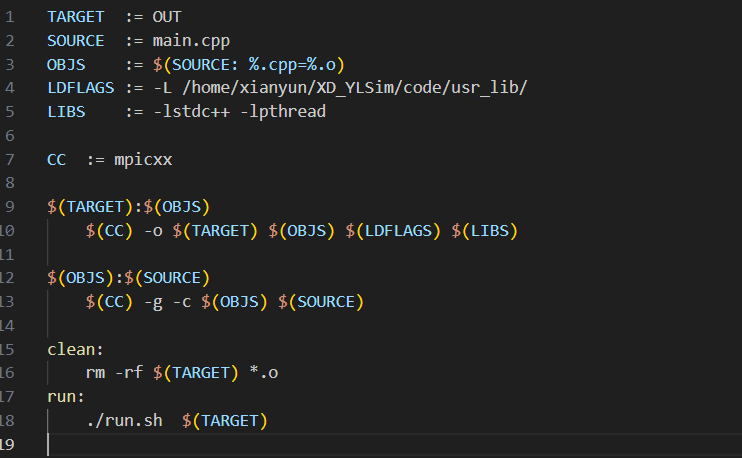
**2，用户仿真模型的编译**

参照旧操作文档户仿真模型的编译一节。

**3，服务器代码的编译与运行**

首先，将要编译的代码放在一个文件夹下。保证三个节点有完全相同的路径名（可能需要手动到三个节点分别创建文件夹）。

在提供的makefile模板基础上，针对要编译的文件，修改makefile。各参数的意义见旧操作文档。另可上网学习makefile如何使用。我们的服务器并行调度框架代码中，使用了c++ std一些模板，以及pthread库用于在linux下使用多线程，因此要加编译选项-lstdc++ -lpthread



命令行键入make,编译代码。如果没有报错信息，说明编译成功。进行下一步。如果有错，请根据报错信息修改代码中的错误。

在主节点可执行文件的当前路径下，寻找servers文件，该文件中写明了各节点的ip地址和各节点要启动的进程数。

如果要在一个节点上运行（常见测试代码的情景），命令行键入：mpirun -np XX ./OUT

其中XX是要启动的并行进程数，OUT是可执行文件名，在makefile中我们把它命名为OUT

如果要在三个节点上运行（常见测试代码的情景），把所需复制的文件手动复制到三个节点，然后命令行键入：mpirun -np 352 -f service ./OUT 三个节点共352个cpu核心，-f service表示mpi启动的进程数在三个节点间的分配方式按service文件中所描述的规则进行。

如果要在三个节点之间正式部署：

命令行键入make run 如上图所示，make run之后会启动shell脚本run.sh，内容如下：



该脚本会将可执行文件和用户自定义的动态库复制到三个节点。因此要注意上一节的提醒：保证三个节点有完全相同的路径名（可能需要手动到三个节点分别创建文件夹）。

该脚本复制完成后，会启动脚本restart.sh，该脚本监控循环监测mpirun进程，如果找不到该进程，说明我们的服务器程序出错挂掉了，重启之。

在使用此种方式部署了程序后，如果要想让程序停止，要先杀掉重启脚本，再杀掉程序。方法如下：命令行键入：ps -ef | grep restart | grep -v grep，得到运行重启脚本的进程的进程号XXX，再键入kill -9 XXX 杀掉该进程；再ps -ef | grep mpirun | grep -v grep，得到运行mpi服务程序的进程的进程号XXX，再键入kill -9 XXX 杀掉该进程。

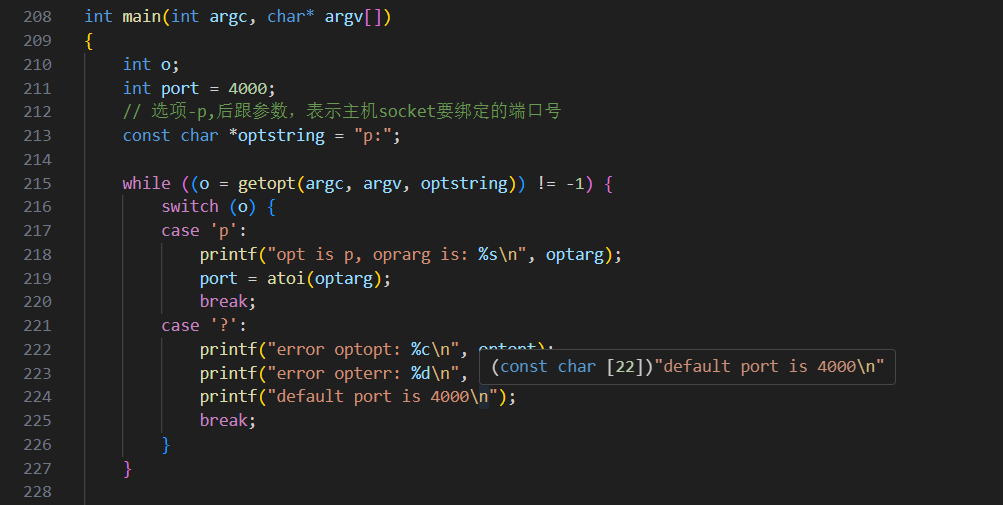
**4，常见debug策略**

有时代码会挂掉，但是mpi多进程代码难以使用gdb调试。因此可以使用core文件调试。设置让程序挂掉后生成core文件，此时再使用gdb调试core文件，找到bug所在。（网上搜索如何让程序生成core文件、如何使用gdb调试core文件）

关于gdb的使用，请参考网上相关教程。在代码开发中，gdb应学会熟练使用。

**5，服务器代码结构**

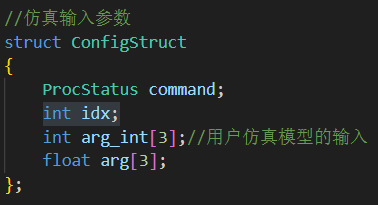
采用这样的并行策略：分为一个主进程（进程号为0）和从进程（除了主进程的其余进程）。主进程负责与上位机的通信，以及调度从进程、分发任务；从进程负责进行任务计算。当从进程空闲时，会通过MPI点对点通信向主进程发送一个计算请求，主进程根据自身状态，如果要计算，并且当前任务池中还有剩余，就发送计算命令帧给某个向他发送计算请求的从进程；任务池计算完成后，发送OVER帧给从进程，等待接收从进程本轮计算的结果数据。 同时从进程发送计算请求后，一直等待主进程给他发消息，收到消息就判断消息类型，如果是计算帧，就进行一次计算（该帧同时携带计算输入配置信息）并把结果存起来，如果是OVER帧，就把本轮计算的结果发给主进程。



Main函数中开头的这一段是使用getopt函数，将命令行输入的main函数参数接进来解析。目前只设置了-p参数用来表示主机TCP socket绑定的端口号。以后可在此基础上拓展。

Sim\_lib.h中写了用户仿真输入、输出接口，以及用户仿真函数调用入口。注意，输入结构体中的ProcStatus command; int idx两项不可更改。输出结构体的int idx;项不可更改。当输入输出结构体修改时，请遵守以下规则：

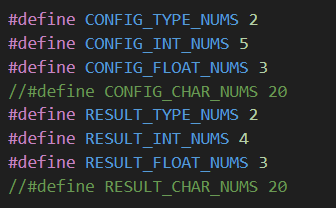
1，出于字节对齐的要求，请将同一类型的结构体成员放在一起，如（枚举类算作int）：



而不是



2，Sim\_lib.h中的宏也要修改



如上文，ConfigStruct结构体中成员有两种类型（int和float），int型有5个（枚举类算作int），float型有3个。

3，mpi\_manage.cpp中的commit\_new\_type函数，







因为ConfigStruct和ResultStruct结构体要在MPI消息中使用，故使用前要在MPI中对其注册。如上图。

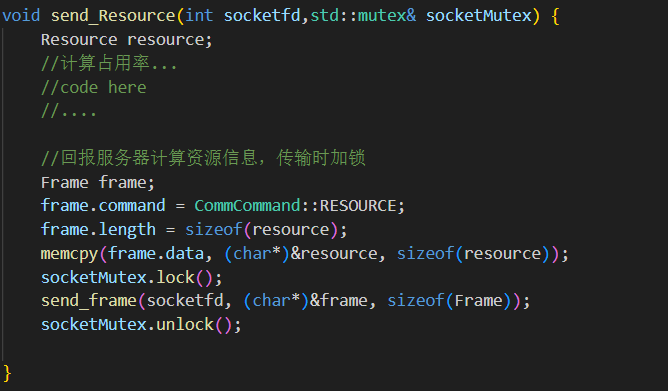
主要注意：

当ConfigStruct或ResultStruct修改后，红线处可能也要修改；

Type类型多了以后，old\_type\_array、blocklens\_array、displs\_array要有old\_type\_array[2]等等；

类型改变（如float变double），MPI\_FLOAT可能要变成MPI\_DOUBLE

另：



回报服务器资源功能的接口留在了这里，请补充。

要避免两个线程同时向同一个socket发送消息，这样可能会有bug，故向服务器回传信息时加了锁（send\_Resource和回传结果recv\_CurRoundAllResults的调用不在一个线程中）