ILCSERVER

version ilcserver-v2.4 and above

The ilcserver is the interface between the run control and the readout program. The program flow is shown in the figure ILCS-1 below. Only one instance of the program is allowed to run at a time, to prevent more than one instance tp run is a file created: LOG_PATH/ilcserver.pid. This file contains the process id of the ilcserver. If this file exist then the server will exit immediately. Log messages are written into the file LOG_PATH/ilcserver-<pid>.log, where pid is the process id. These two files are removed when the server ends normally. In case of a program crash they are not removed. The process id file ilcserver.pid must then be removed before the program can start again. The definition of LOG_PATH is in the source code of ilcserver.cxx. The ilcserver is shutdown when receiving the unix signals: SIGINT, SIGTERM, or SIGQUIT. It will try to kill the readout program by sending a unix signal SIGTERM to it, and it closes all network connections.

If there is no other ilcserver found it forks itself, and the forked instance executes the readout program defined by PROG_PATH (defined in ilcserver.cxx). If this fails a message is written to standard error, and the program exits. Some of the arguments for the readout program is set in the command.

The configuration file, ilcserver.cfg in the working directory, is read.

- SERVER.READOUT = readout server port
- SERVER.LOCAL = local run control server port
- SERVER.GLOBAL = global run control server port

These are the three network ports the ilcserver is listening on. It listens for a connection from the readout program, the local run control, and an optional second (e.g. global) run control. The ilcserver can only handle two connections, intended for the readout and one run control. *There is not yet defined any second run control handling.*

- CLIENT.LOCAL = port used to connect to the local run control server
- CLIENT.READOUT = port used to connect to the readout server

After having received *CLI from the readout program and *SRV from the local run control, then the ilcserver opens a connection to them as a client using these ports. *There is not yet defined any second run control handling.* It use the IP-number from which the readout/control program made the first connection.

A data packet is received is decoded depending on the packet source:

- decodeServerPort1 : decodes the packet from the local run control, and acts depending on the content. The main task is to send it further to the readout server (CLIENT.READOUT).
- decodeServerPort2 : a dummy routine since no second run control is yet defined. The purpose of this routine will be to interface between it's run control format and the readout server format.
- decodeServerPort3: decodes the packet from the readout program, and acts depending on the content. It encodes it to the format of the local run control (CLIENT.READOUT), in this case the format is the same as received from the readout, or to the second yet undefined run control.

The communication protocol with the local run control is described in the local run control document.

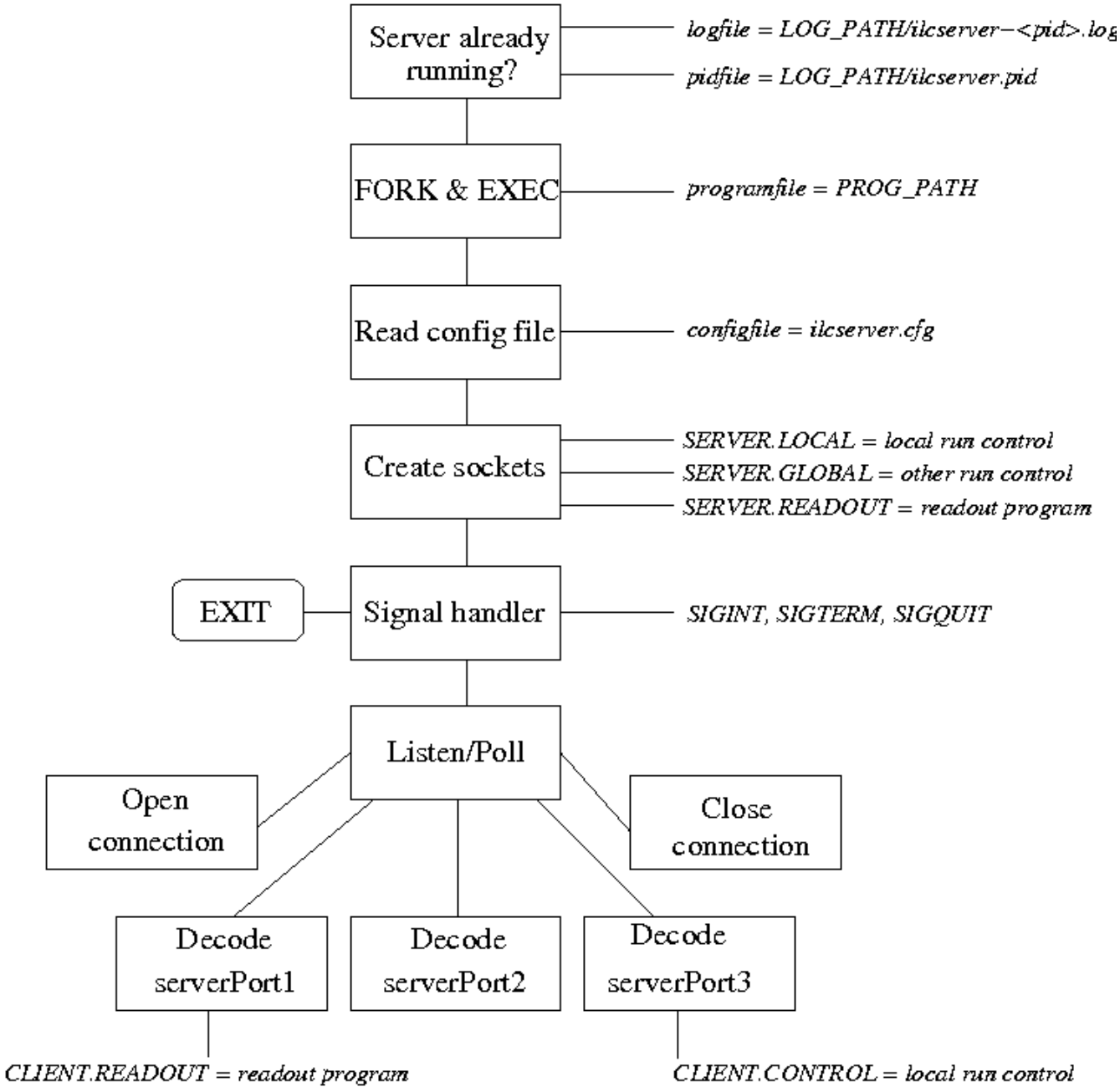

Figure ILCS-1: The program flow

Files

ilcserver.cxx	main source code
ilcserver.cfg	configuration file
JMonSocket.cxx	network routines
JMonSocket.h	include file for network routines
readme.txt	updates made
Makefile	compile and link
LOG_PATH/ilcserver.pid	created when program running, content is processid
LOG_PATH/ilcserver-<pid>.log	logfile

COMPILE AND RUN

make

./ilcserver