

MONITORING

The monitoring system is briefly shown in figure MON-1. The readout starts three threads, the raw event readout, and two monitoring threads. The readout is storing an event in the shared physical memory, this event is accessed by the online monitoring thread using a locking mechanism to avoid the readout to overwrite an event currently sent by the monitoring. The playback thread is reading a file, independent of the physical event buffer. An event is sent on request from the monitoring task.

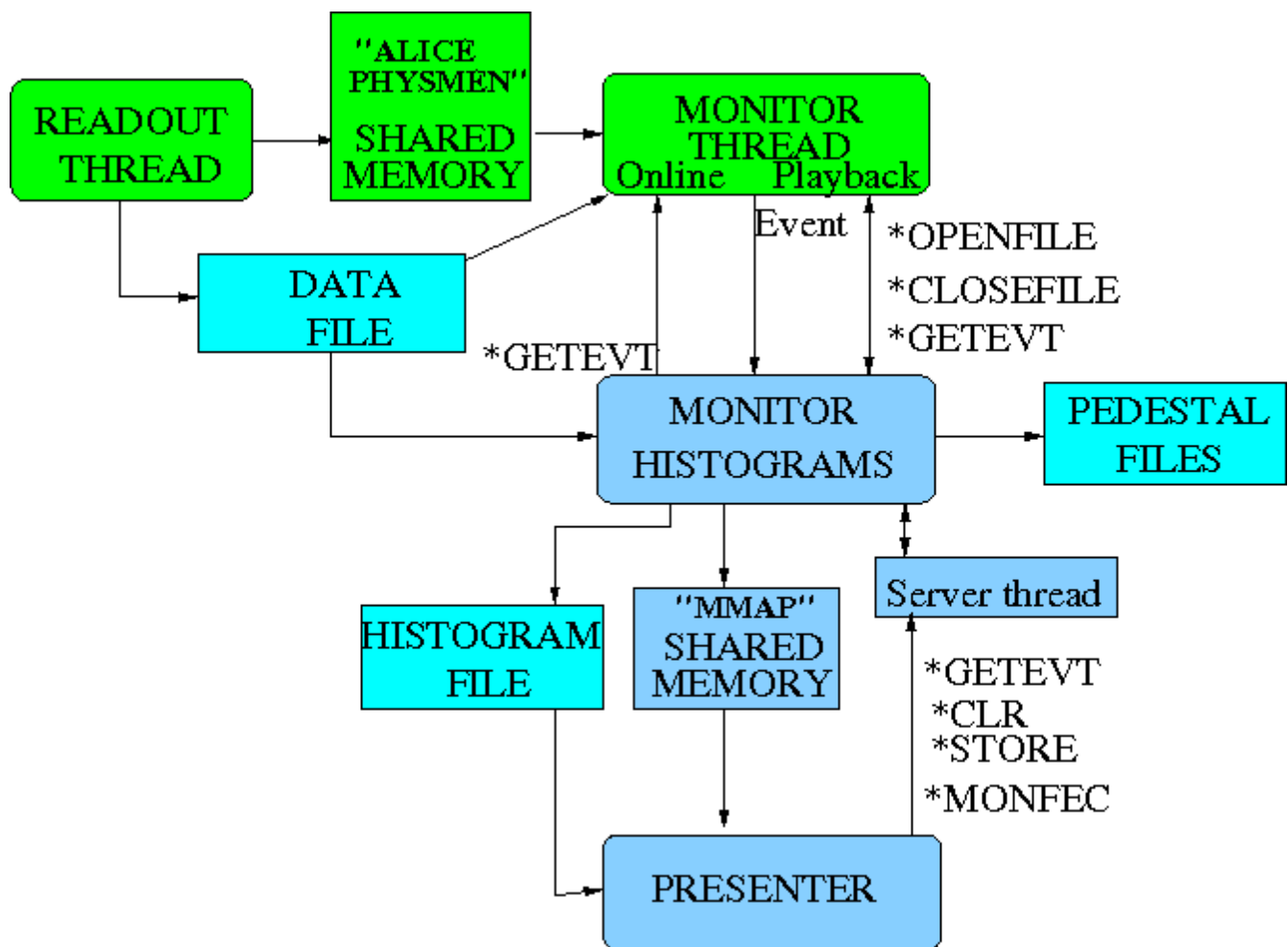


Figure MON-1: Brief overview of monitoring system

The communication between the monitoring threads and the monitoring program is through network commands, sent from the monitoring task.

- | | |
|-----------------------------------|---|
| *OPENFILE <filename> | open file for playback |
| *CLOSEFILE | close playback file |
| *GETEVT | get an event, the event is sent though the network. |

The event handling loop in the readout is shown in figure MON-2. The monitoring threads listen to the network for requests. The online monitoring thread is using the built in locking mechanism in pthread to have exclusive usage of the event buffer. No such mechanism is needed for playback. If there is no run active is the monitoring thread sending back a data buffer with a status message instead of a true event.

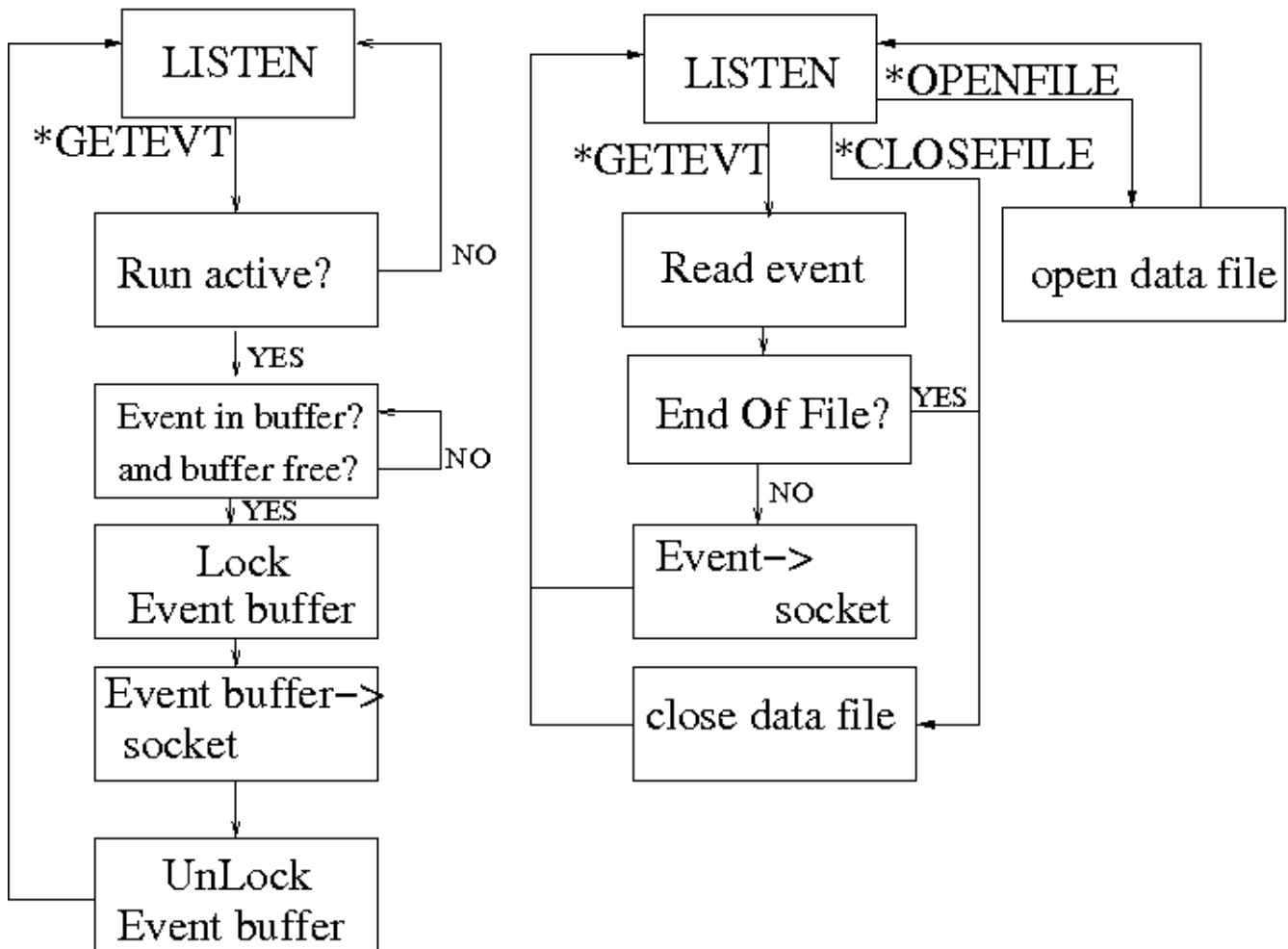


Figure MON-2: Event monitoring loop

The monitoring task can also read events from a data file directly. It fills histograms and stores them in a ROOT format in a file (online.root) and in mmap shared memory for access by the histogram presenter. The communication with the presenter is described in the document of the presenter. The monitor task starts up a thread for listening for commands from the presenter.

In a special running mode can the monitoring task be used to read a raw data file and produce pedestal files.

The monitoring program understands the command line arguments:

-i, --infile : input file in the input directory
 -I, --indir : input data directory overrides default (DATA_DIR in monitor.h)
 -O, --outdir : output data directory overrides default(ROOT_DIR in monitor.h)
 -p, --playback : get event from monitor server in playback mode
 -a, --all : change file when EOF encountered in playback mode
 -s, --server : monitor server IP address
 -e, --events : make rootfile for each event (only when reading data file)
 -P, --pport : port used for communication with presenter
 -w, --wait : wait for a run from online server in seconds - then retry
 wait for presenter to request an event when reading a file

--wait 0/1 if reading file, 1 = wait for *GETEVT from histogram presenter to read next event

FILES

monitor.cxx, monitor.h	main program
JPed.cxx,JPed.h	pedestal calculations
JMonSocket.cxx,JMonSocket.h	network connection with monitor thread
JMonHist.cxx,JMonHist.h	histogramming routines
JMonFile.cxx,JMonFile.h	reading of raw data file
JMonEvent.cxx,JMonEvent.h	unpack an event
JMonDisplay.cxx,JMonDisplay.h	communication with presenter
Makefile	compile and link
config.dat	which histograms to fill – read at program start
histos.dyn	a few histogram where limits can be changed dynamically
mapchvsxy.txt	mapping of channels to x,y coordinates
mon-file.sh	script to monitor with events read from a data file
mon-online.sh	script to monitor with events from online readout
mon-ped.sh	script to run monitor to make pedestal files from a data file
readme.txt	changes done