Status of the data process in commissioning with beam

Genki Nukazuka (RIKEN/RBRC)

Motivation

After taking data with the beam, we processed data by hand:

- Decoding to make hit-wise TTree (Joseph's program)
- Decoding to make event-wise TTree (Takashi's program)
- Making ADC and channel distribution plots (Misaki's program)

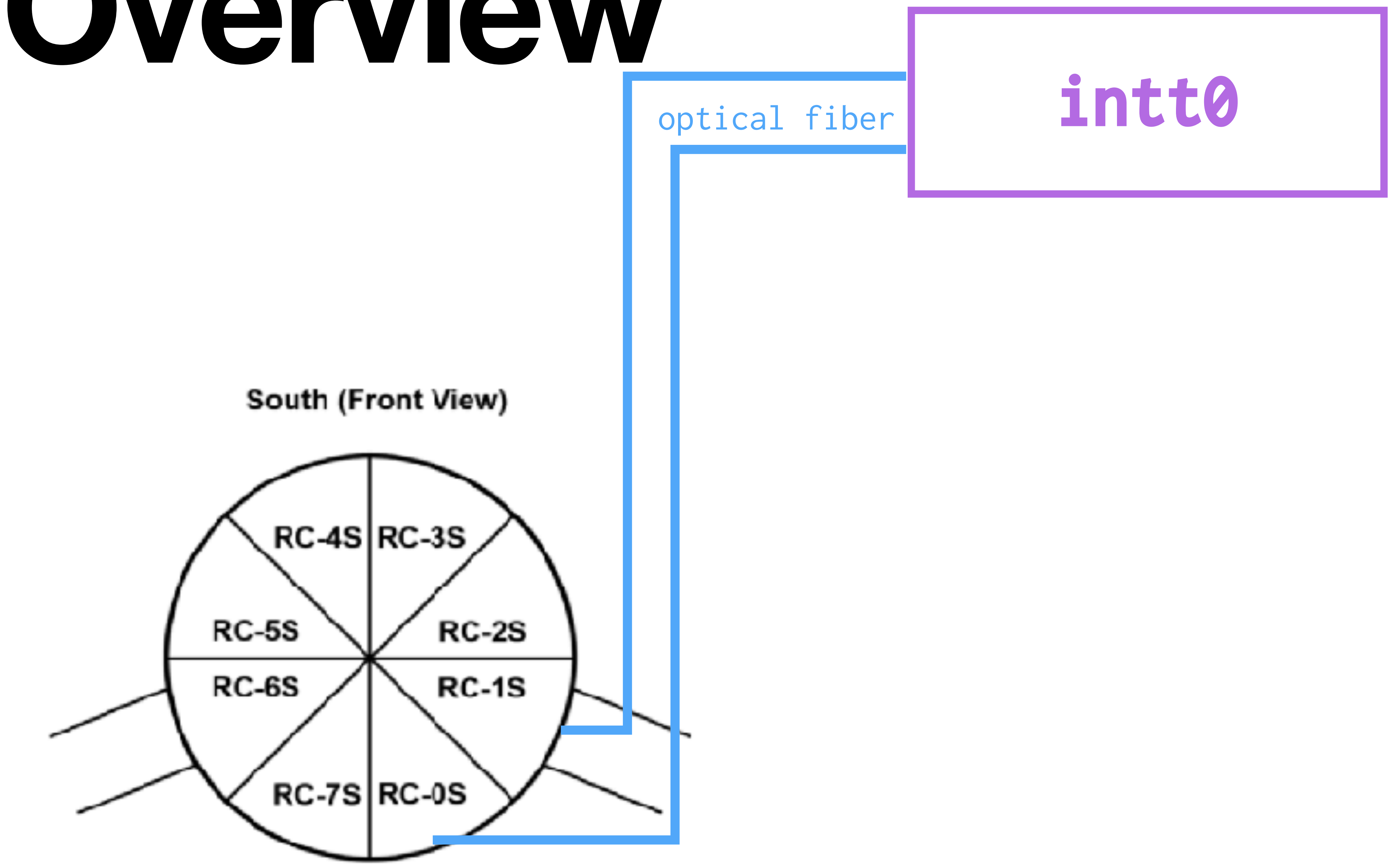
It's trouble, so I introduced macros to do them in one line.

But it's still trouble. It has to be automated.
Additionally, I'll be off-site from June/24 — July/21(?), automation is crucial.
→ I made it. It's under testing.

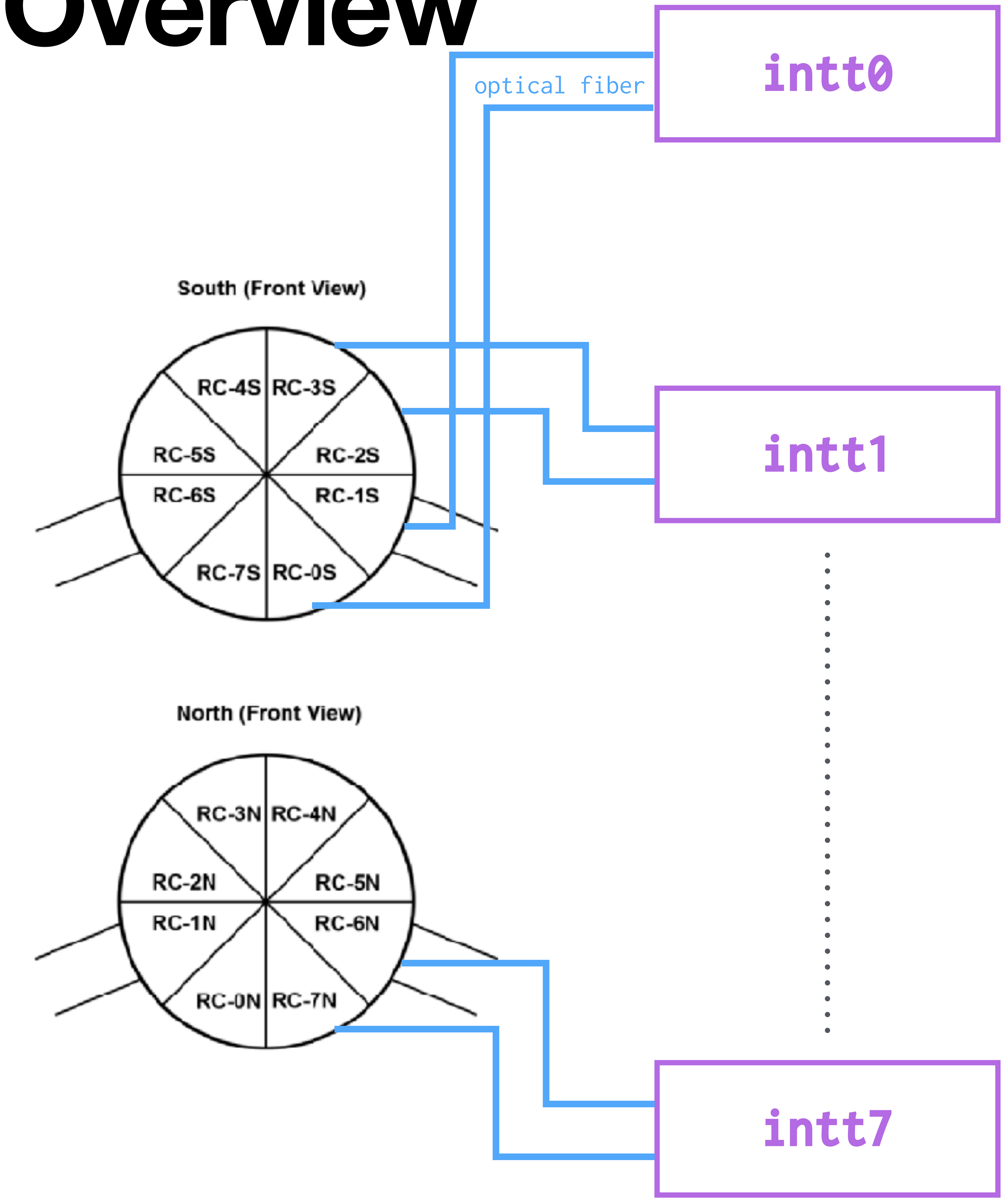
On the other hand, those processes should be done in SDCC servers not to affect DAQ. Milan is working on it.

∴ My automated data process is fine for now. We can rely on it for a few weeks.

Overview

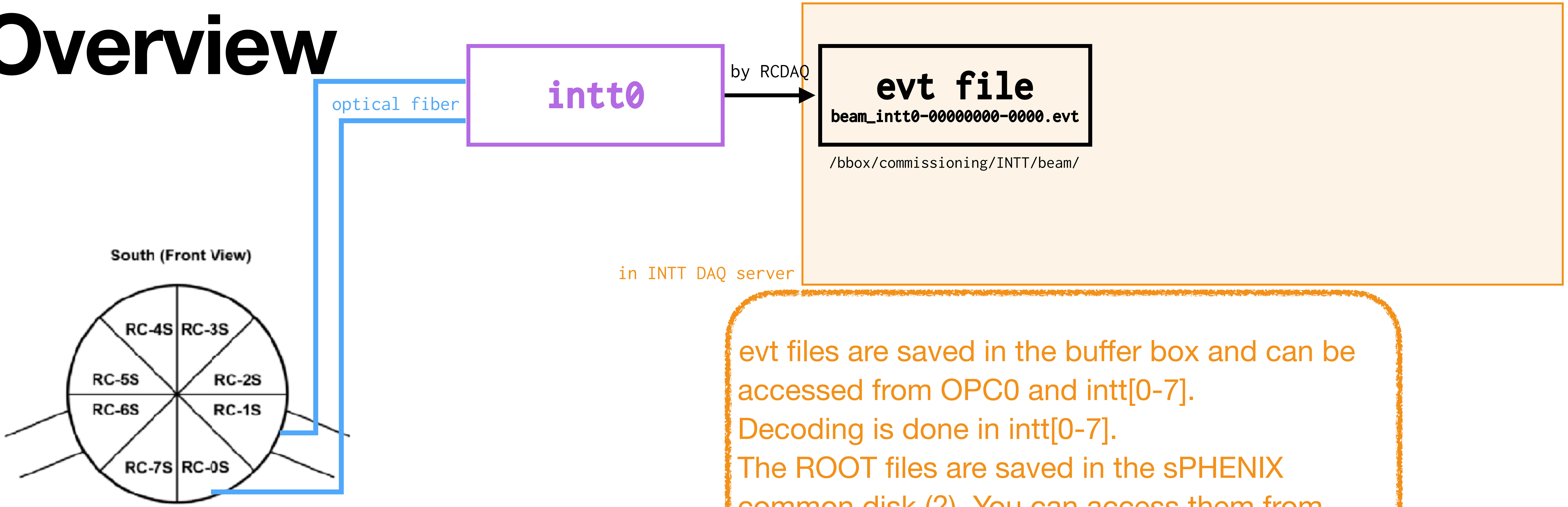

intt0
optical fiber
South (Front View)
RC-4S
RC-3S
RC-5S
RC-2S
RC-6S
RC-1S
RC-7S
RC-0S

Overview



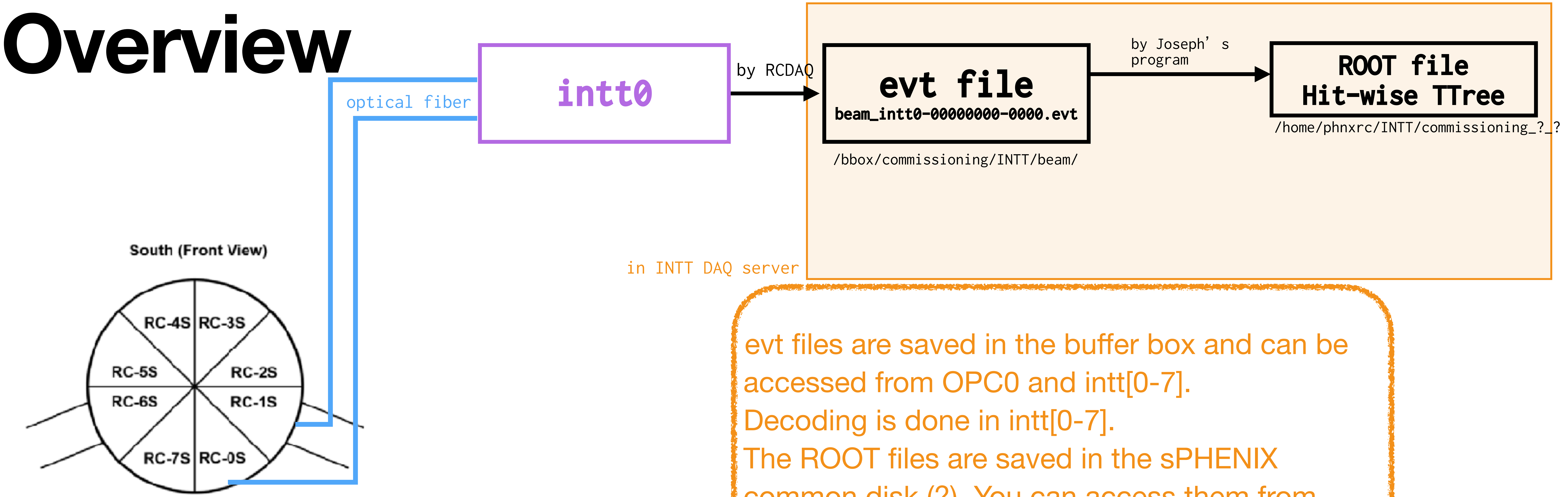
optical fiber
intt0
intt1
intt7
South (Front View)
RC-4S
RC-3S
RC-5S
RC-2S
RC-6S
RC-1S
RC-7S
RC-0S
North (Front View)
RC-3N
RC-4N
RC-2N
RC-5N
RC-1N
RC-6N
RC-0N
RC-7N

Overview



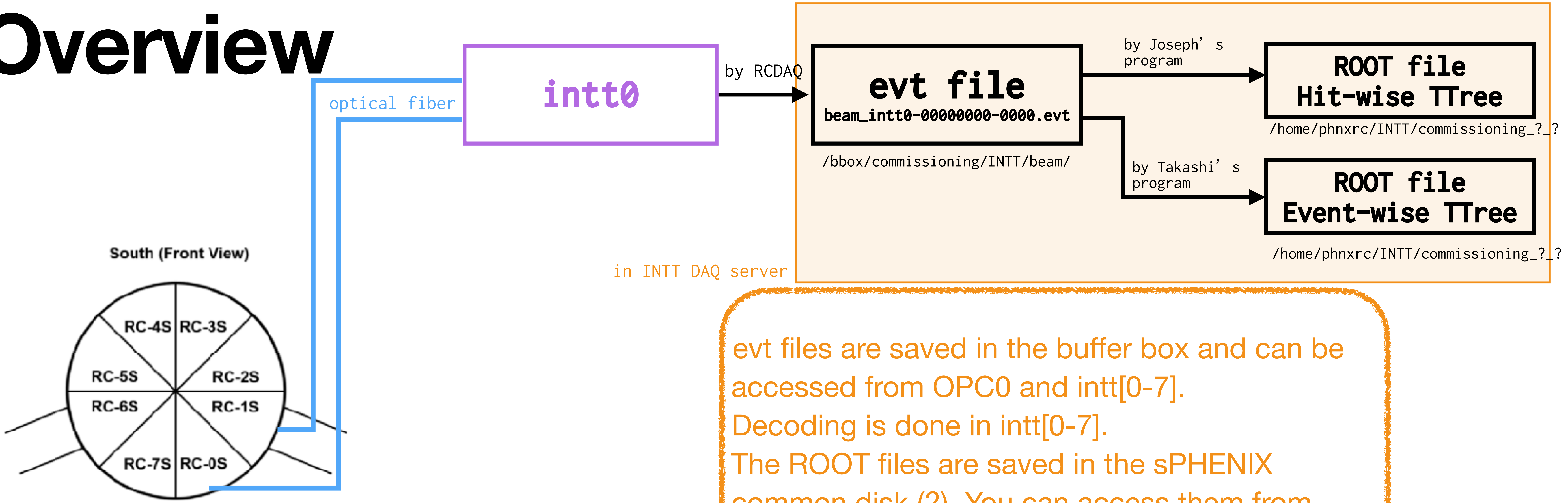
evt files are saved in the buffer box and can be accessed from OPC0 and intt[0-7].
Decoding is done in intt[0-7].
The ROOT files are saved in the sPHENIX common disk (?). You can access them from OPC0 and intt[0-7], and inttdaq.

Overview



evt files are saved in the buffer box and can be accessed from OPC0 and intt[0-7].
Decoding is done in intt[0-7].
The ROOT files are saved in the sPHENIX common disk (?). You can access them from OPC0 and intt[0-7], and inttdaq.

Overview



evt files are saved in the buffer box and can be accessed from OPC0 and intt[0-7].
Decoding is done in intt[0-7].
The ROOT files are saved in the sPHENIX common disk (?). You can access them from OPC0 and intt[0-7], and inttdaq.

Overview

South (Front View)
RC-4S
RC-3S
RC-5S
RC-2S
RC-6S
RC-1S
RC-7S
RC-0S
optical fiber
intt0
by RCDAQ
evt file
beam_intt0-00000000-0000.evt
/bbox/commissioning/INTT/beam/
by Joseph’ s program
ROOT file Hit-wise TTree
/home/phnxrc/INTT/commissioning_?_?
by Takashi’ s program
ROOT file Event-wise TTree
/home/phnxrc/INTT/commissioning_?_?
in INTT DAQ server
by Misaki’ s program
inttdev@inttdaq:
/home/inttdev/INTT/data/commissioning_?_?
in inttdaq

evt files are saved in the buffer box and can be accessed from OPC0 and intt[0-7].
Decoding is done in intt[0-7].
The ROOT files are saved in the sPHENIX common disk (?). You can access them from OPC0 and intt[0-7], and inttdaq.

ADC and channel distribution plots are generated in inttdaq.
Plots are stored in inttdaq.

Overview

optical fiber
intt0
by RCDAQ
evt file
beam_intt0-00000000-0000.evt
/bbox/commissioning/INTT/beam/
by Joseph' s program
ROOT file Hit-wise TTree
/home/phnxrc/INTT/commissioning_?_?
by Takashi' s program
ROOT file Event-wise TTree
/home/phnxrc/INTT/commissioning_?_?
by Misaki' s program
inttdev@inttdaq: /home/inttdev/INTT/data/commissioning_?_?
South (Front View)
RC-4S
RC-3S
RC-5S
RC-2S
RC-6S
RC-1S
RC-7S
RC-0S
in INTT DAQ server
intt1
by RCDAQ
evt file
beam_intt0-00000000-0000.evt
/bbox/commissioning/INTT/beam/
by Joseph' s program
ROOT file Hit-wise TTree
/home/phnxrc/INTT/commissioning_?_?
by Takashi' s program
ROOT file Event-wise TTree
/home/phnxrc/INTT/commissioning_?_?
by Misaki' s program
inttdev@inttdaq: /home/inttdev/INTT/data/commissioning_?_?
North (Front View)
RC-3N
RC-4N
RC-2N
RC-5N
RC-1N
RC-6N
RC-0N
RC-7N
intt7
by RCDAQ
evt file
beam_intt0-00000000-0000.evt
/bbox/commissioning/INTT/beam/
by Joseph' s program
ROOT file Hit-wise TTree
/home/phnxrc/INTT/commissioning_?_?
by Takashi' s program
ROOT file Event-wise TTree
/home/phnxrc/INTT/commissioning_?_?
by Misaki' s program
inttdev@inttdaq: /home/inttdev/INTT/data/commissioning_?_?
in inttdaq

Single process chain

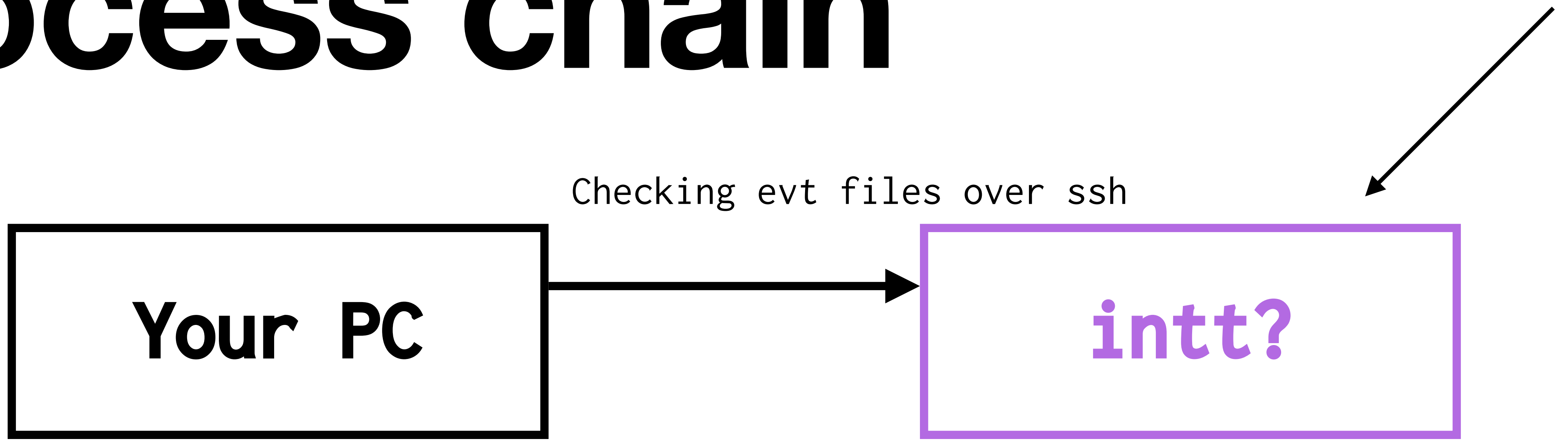
1. Check evt files

`Your PC`

Single process chain

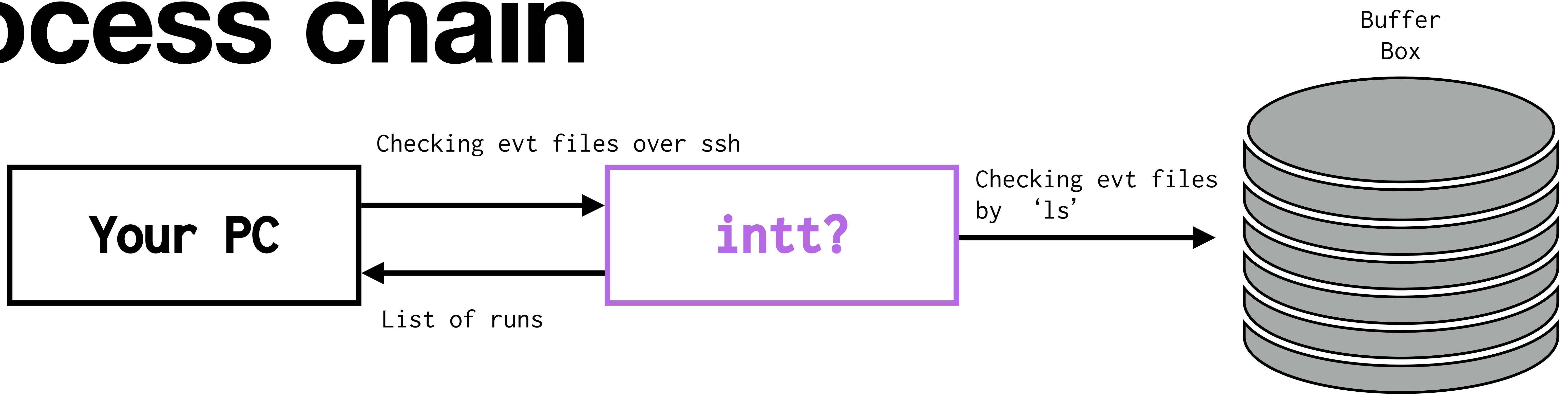
It means one of INTT DAQ server

1. Check evt files



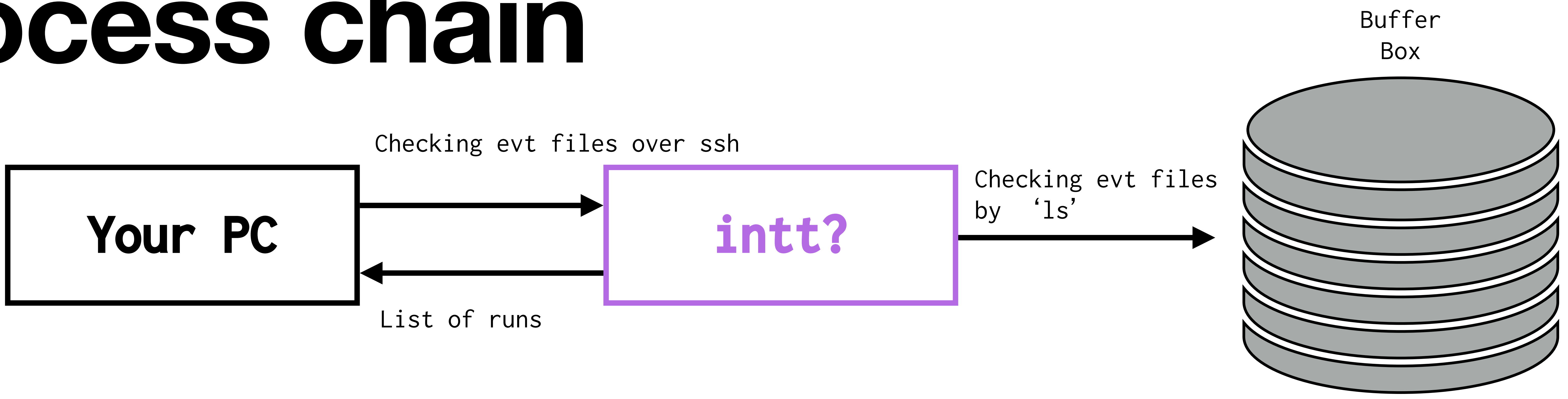
Single process chain

1. Check evt files

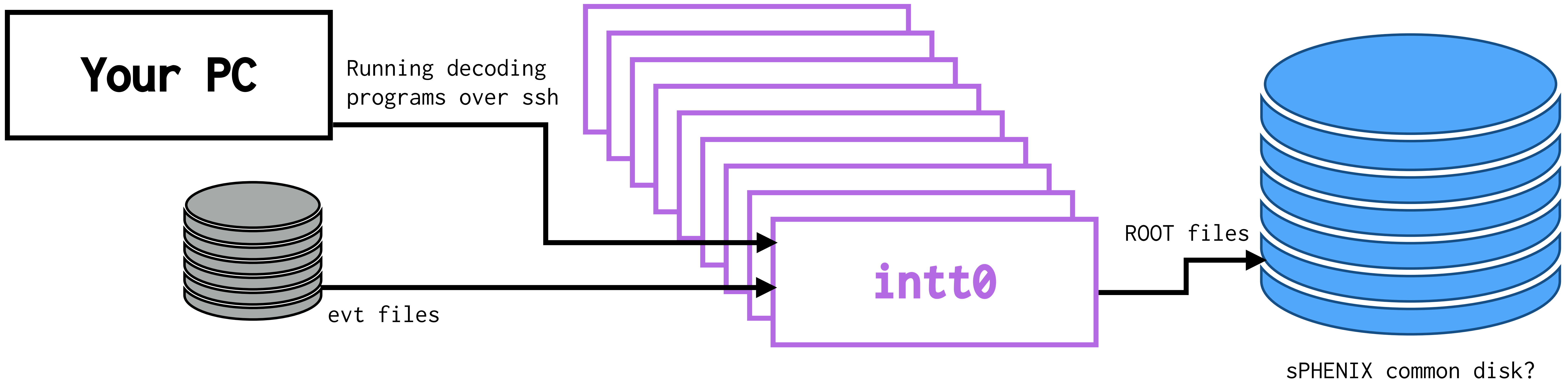


Single process chain

1. Check evt files

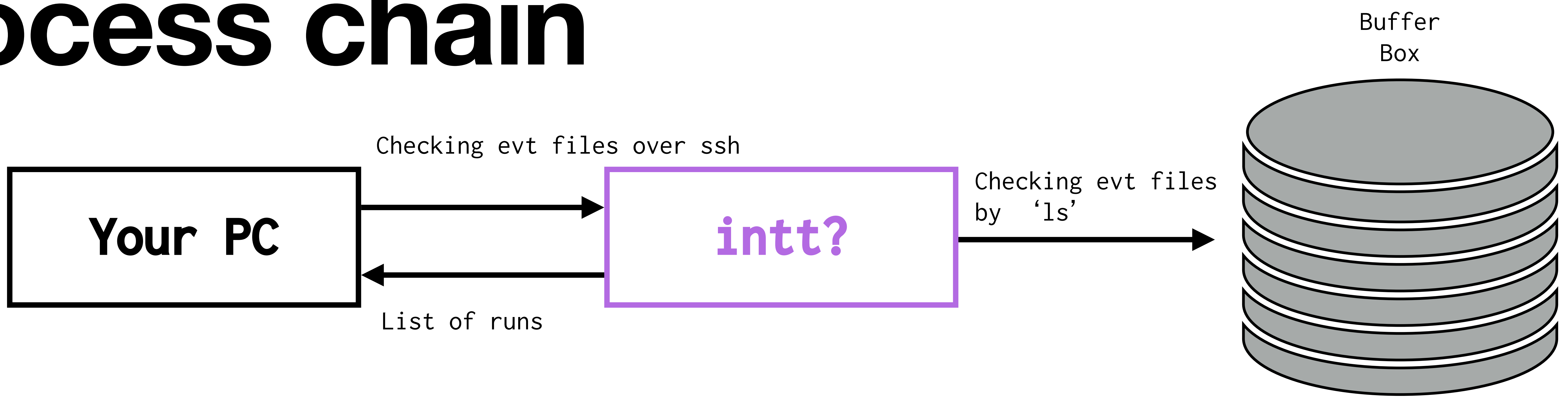


2. Decode evt files

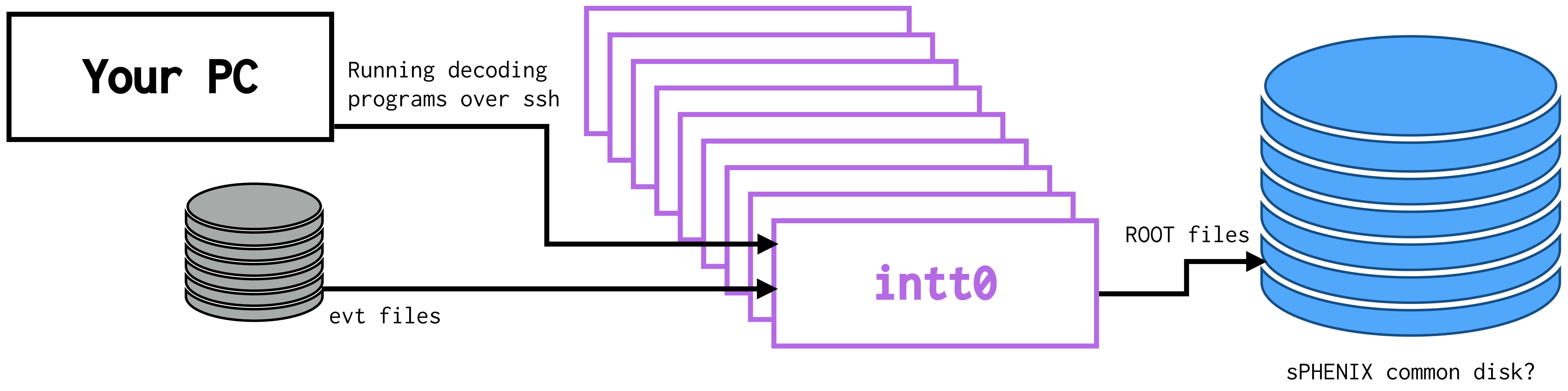


Single process chain

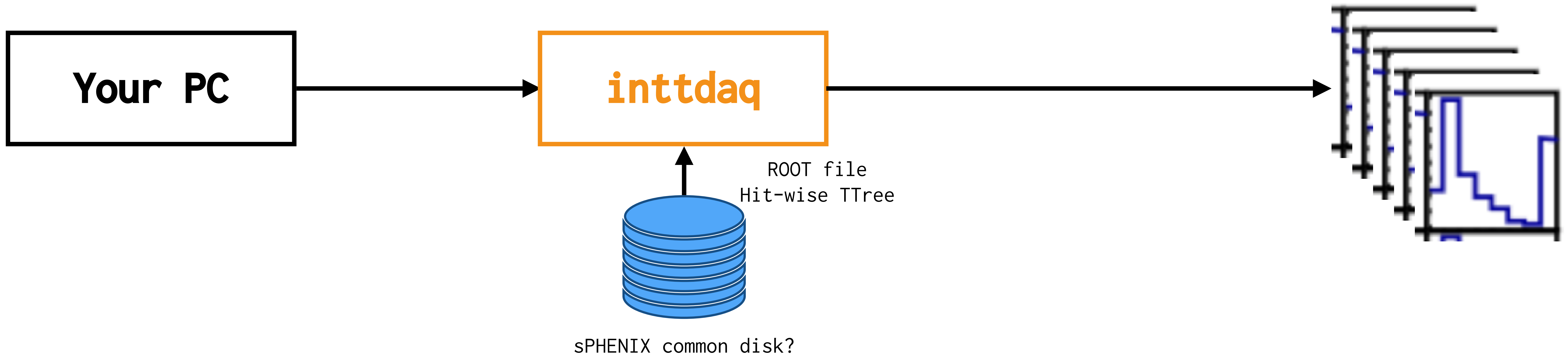
1. Check evt files



2. Decode evt files



3. Make plots



The scripts

I made 2 files for the automated processes:

- process_commissioning_data.py: It does everything.
- process_data: A shell script to be put under $PATH directory so that you can use it like a command

Both can be found on GitHub: [INTT/general_codes/genki]

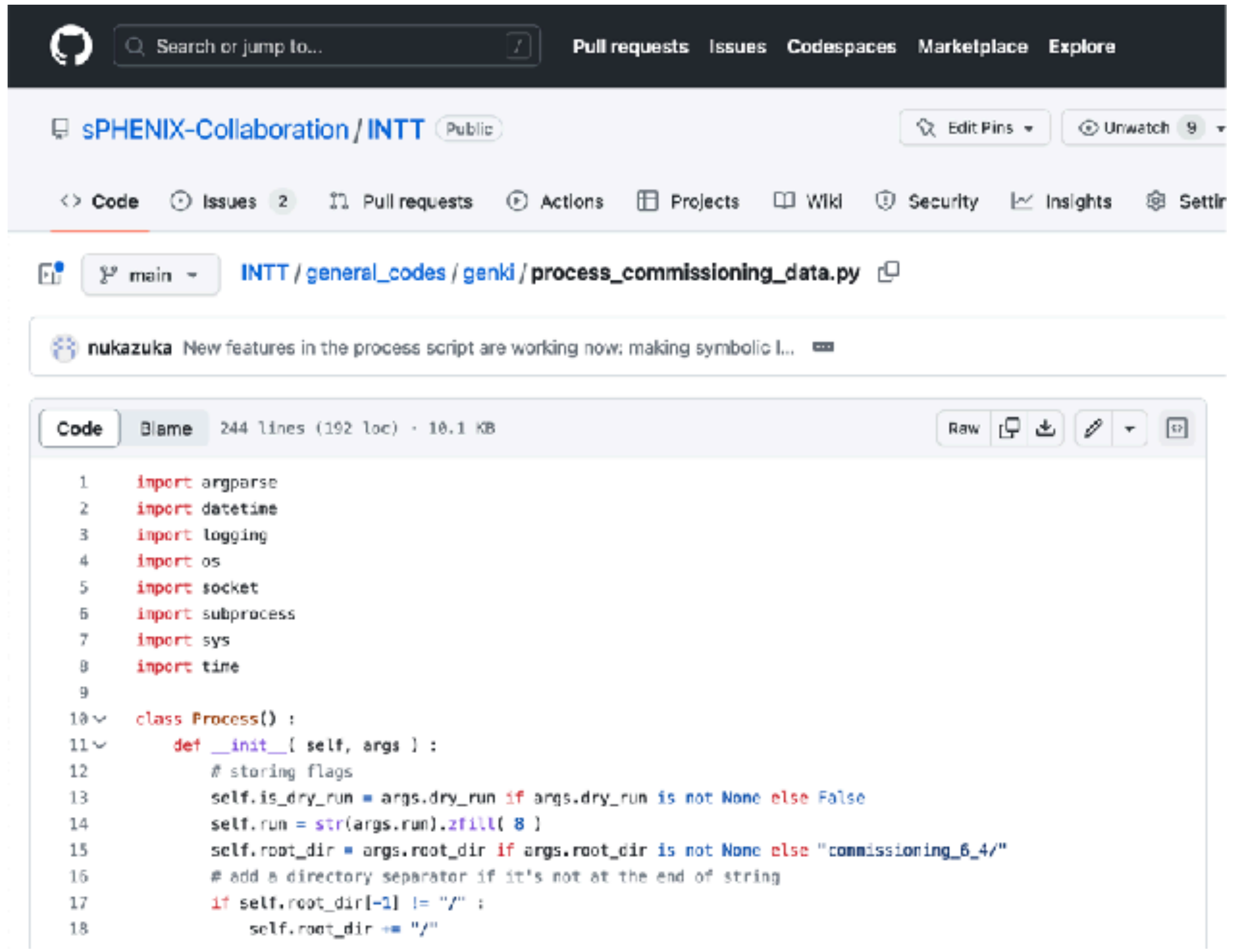
The scripts are set up in inttdaq, Genki's Mac, and Genki's mini-desktop.
If you want, you can set them up in your PC.

Set up

- SSH configuration needs to be done same as my assumption. (It's not good)
- Python3 required

Usage

- python3 process_commissioning_data [options] run_number
  or
- process_data [options] run_number
- run_number is mandatory.



INTT / general_codes / genki / process_commissioning_data.py

nukazuka New features in the process script are working now: making symbolic l...

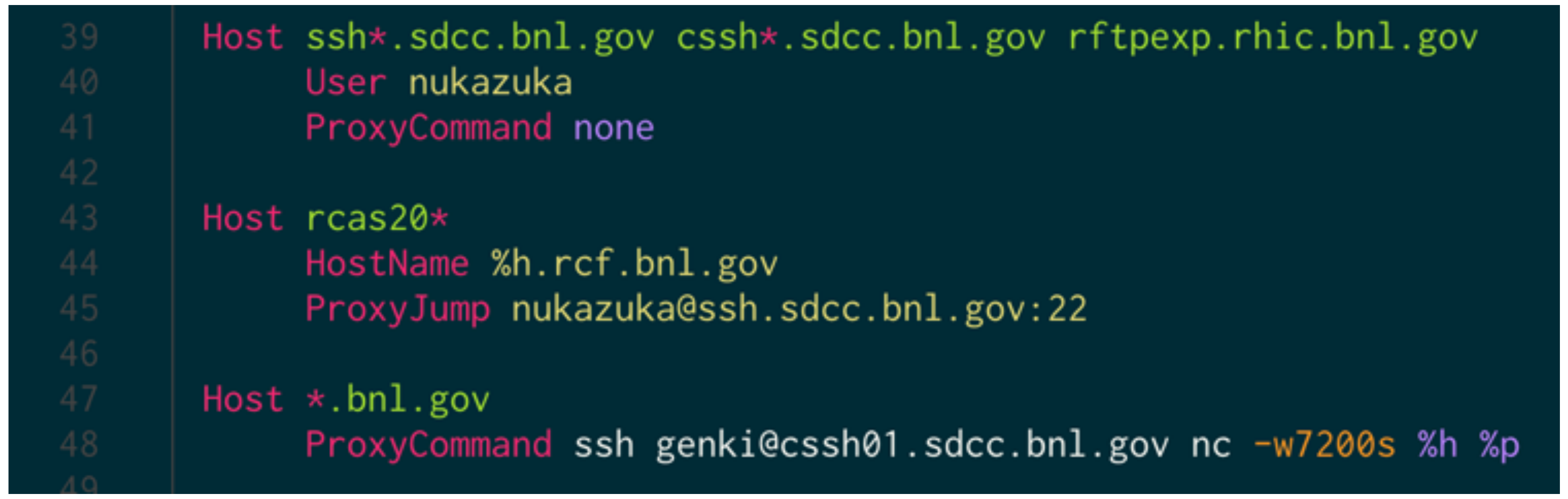
244 lines (192 loc) · 10.1 KB

```
import argparse
import datetime
import logging
import os
import socket
import subprocess
import sys
import time

class Process() :
    def __init__( self, args ) :
        # storing flags
        self.is_dry_run = args.dry_run if args.dry_run is not None else False
        self.run = str(args.run).zfill( 8 )
        self.root_dir = args.root_dir if args.root_dir is not None else "commissioning_6_4/"
        # add a directory separator if it's not at the end of string
        if self.root_dir[-1] != "/" :
            self.root_dir += "/"
```

The scripts: SSH setting

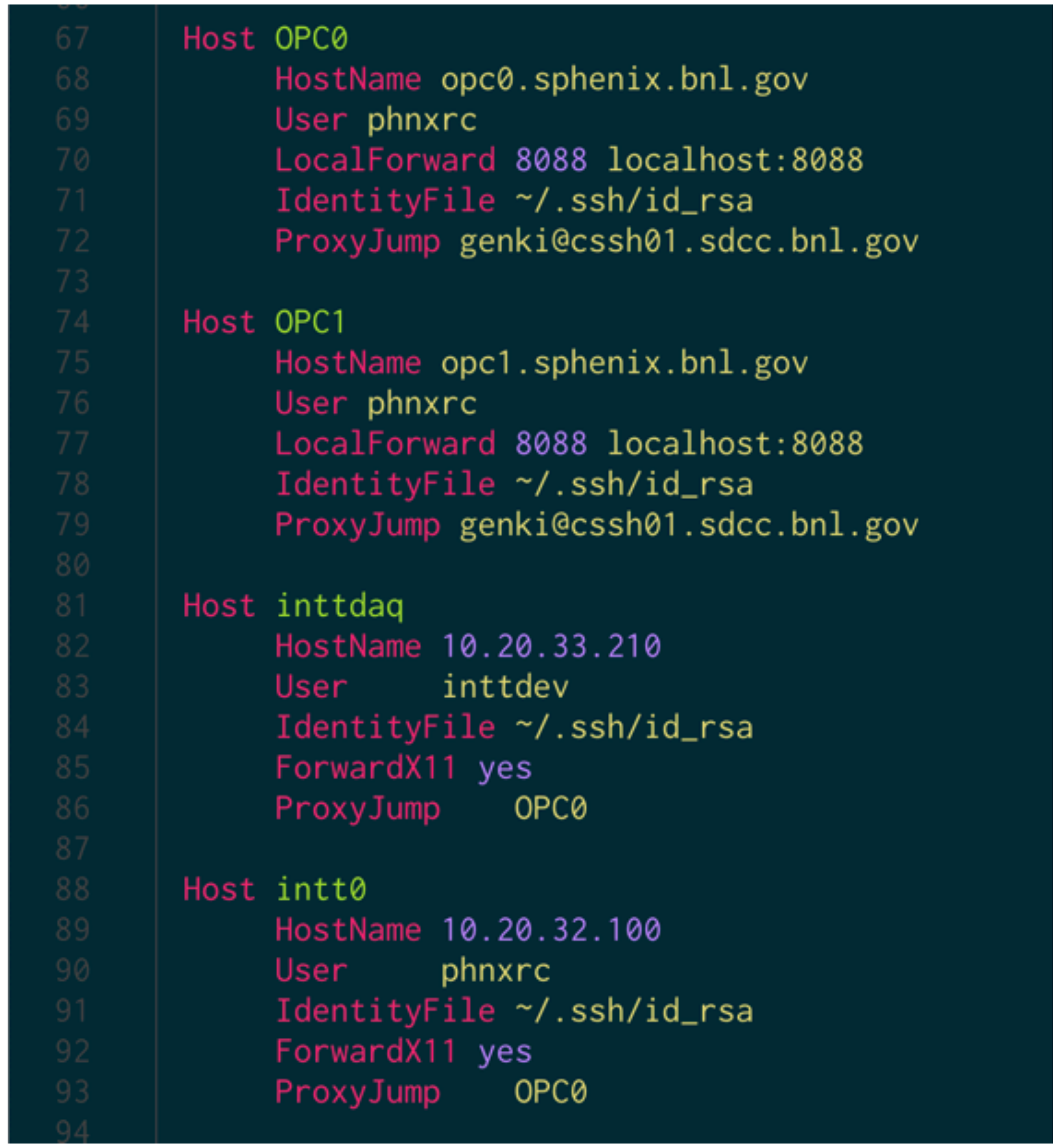
I assumed users can log in to some servers with SSH key:

- intt0, intt1, …, intt7
- inttdaq



```
Host ssh*.sdcc.bnl.gov cssh*.sdcc.bnl.gov rftpexp.rhic.bnl.gov
     User nukazuka
     ProxyCommand none

Host rcas20*
     HostName %h.rcf.bnl.gov
     ProxyJump nukazuka@ssh.sdcc.bnl.gov:22

Host *.bnl.gov
     ProxyCommand ssh genki@cssh01.sdcc.bnl.gov nc -w7200s %h %p
```

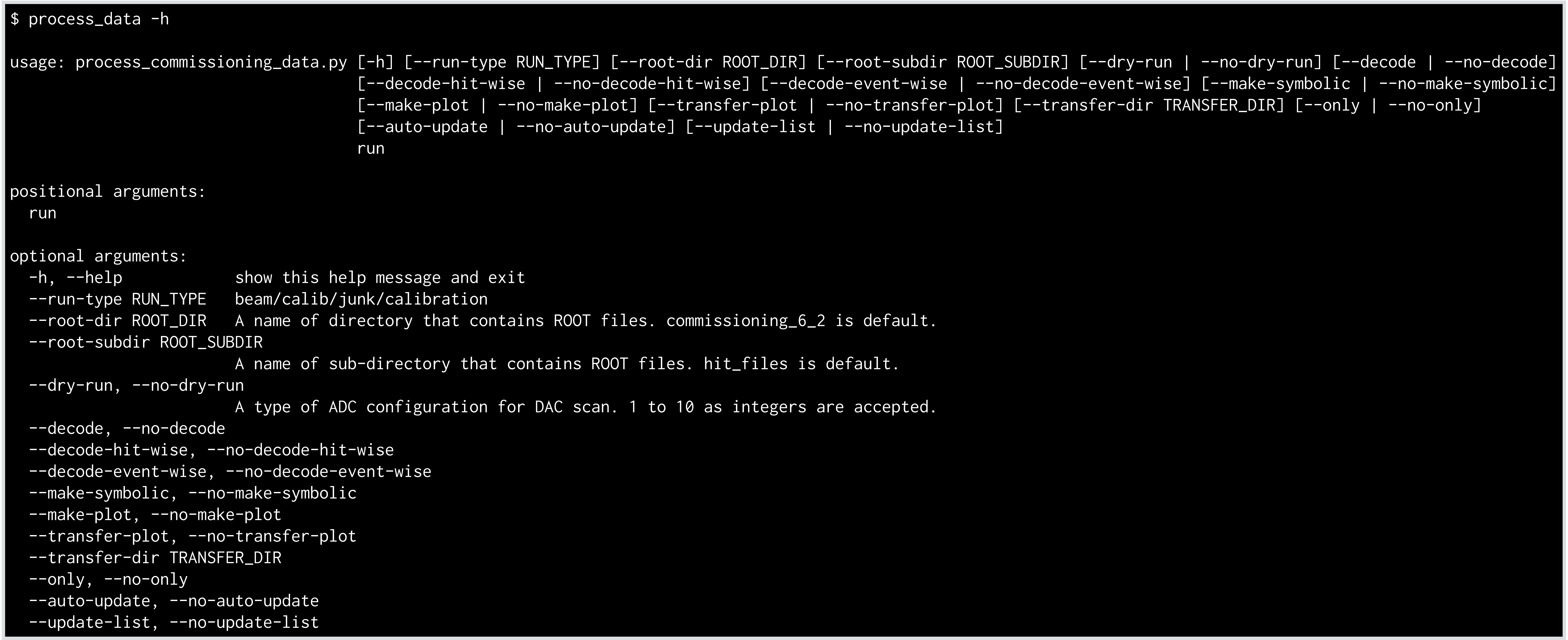
Example



```
Host OPC0
     HostName opc0.sphenix.bnl.gov
     User phnxrc
     LocalForward 8088 localhost:8088
     IdentityFile ~/.ssh/id_rsa
     ProxyJump genki@cssh01.sdcc.bnl.gov

Host OPC1
     HostName opc1.sphenix.bnl.gov
     User phnxrc
     LocalForward 8088 localhost:8088
     IdentityFile ~/.ssh/id_rsa
     ProxyJump genki@cssh01.sdcc.bnl.gov

Host inttdaq
     HostName 10.20.33.210
     User     inttdev
     IdentityFile ~/.ssh/id_rsa
     ForwardX11 yes
     ProxyJump    OPC0

Host intt0
     HostName 10.20.32.100
     User     phnxrc
     IdentityFile ~/.ssh/id_rsa
     ForwardX11 yes
     ProxyJump    OPC0
```

The scripts

If you execute it by yourself, just in case…



```
$ process_data -h

usage: process_commissioning_data.py [-h] [--run-type RUN_TYPE] [--root-dir ROOT_DIR] [--root-subdir ROOT_SUBDIR] [--dry-run | --no-dry-run] [--decode | --no-decode]
                                     [--decode-hit-wise | --no-decode-hit-wise] [--decode-event-wise | --no-decode-event-wise] [--make-symbolic | --no-make-symbolic]
                                     [--make-plot | --no-make-plot] [--transfer-plot | --no-transfer-plot] [--transfer-dir TRANSFER_DIR] [--only | --no-only]
                                     [--auto-update | --no-auto-update] [--update-list | --no-update-list]
                                     run

positional arguments:
  run

optional arguments:
  -h, --help            show this help message and exit
  --run-type RUN_TYPE   beam/calib/junk/calibration
  --root-dir ROOT_DIR   A name of directory that contains ROOT files. commissioning_6_2 is default.
  --root-subdir ROOT_SUBDIR
                        A name of sub-directory that contains ROOT files. hit_files is default.
  --dry-run, --no-dry-run
                        A type of ADC configuration for DAC scan. 1 to 10 as integers are accepted.
  --decode, --no-decode
  --decode-hit-wise, --no-decode-hit-wise
  --decode-event-wise, --no-decode-event-wise
  --make-symbolic, --no-make-symbolic
  --make-plot, --no-make-plot
  --transfer-plot, --no-transfer-plot
  --transfer-dir TRANSFER_DIR
  --only, --no-only
  --auto-update, --no-auto-update
  --update-list, --no-update-list
```

The help document is under construction.

README and E-Log entry will be written too.

The scripts, examples

If you execute it by yourself, just in case…

```
Showing the help message
$ process_data -h

Processing run00000123 (decoding for hit/event-wise TTree, making links in inttdaq, making plots, scp plots to your local env)
$ process_data 123

Test of the process for run00000123, but nothing done
$ process_data --dry-run 123

Runs newly found in the buffer box are processed.
(The first execution doesn’ t work because there is no list for the previous condition. Do twice)
A dummy run number is needed.
$ process_data --auto-update 0

Processing run00000456 (event-wise TTree is not made, making links in inttdaq, making plots, scp plots to your local env)
$ process_data --no-decode-event 0456

Making plots of run00098765 in inttdaq
$ process_data --only --make_plot 98765

Downloading plots of run00078901 from inttdaq to your local env (It doesn’ t care whether they exist or not)
$ process_data --only --transfer-plot 00078901
```

Periodical execution

The script is automatically executed in inttdaq by cron every 10 minutes.

```
*/10 * * * * /home/inttdev/bin/process_data --no-transfer-plot --auto-update  0 2>&1  >> /home/inttdev/INTT/log/inttdaq_cron/`date +\%Y\%m\%d_\%H\%M\%S`.log
```

Runs to be processed

- New runs found in the latest 400 evt files.
  - The last run that appears in the run list is not processed because the run may be ongoing.
  - 400 files / 8 servers < 50 runs are checked. (A run generates more than 8 files depending on data size)
  - If >50 runs are launched in 10 min, some runs are not processed. I think it's safe enough.

Exclusive operation

- If processes with the same name are already running, nothing is done. Otherwise, resources in intt[0-7] are consumed a lot.

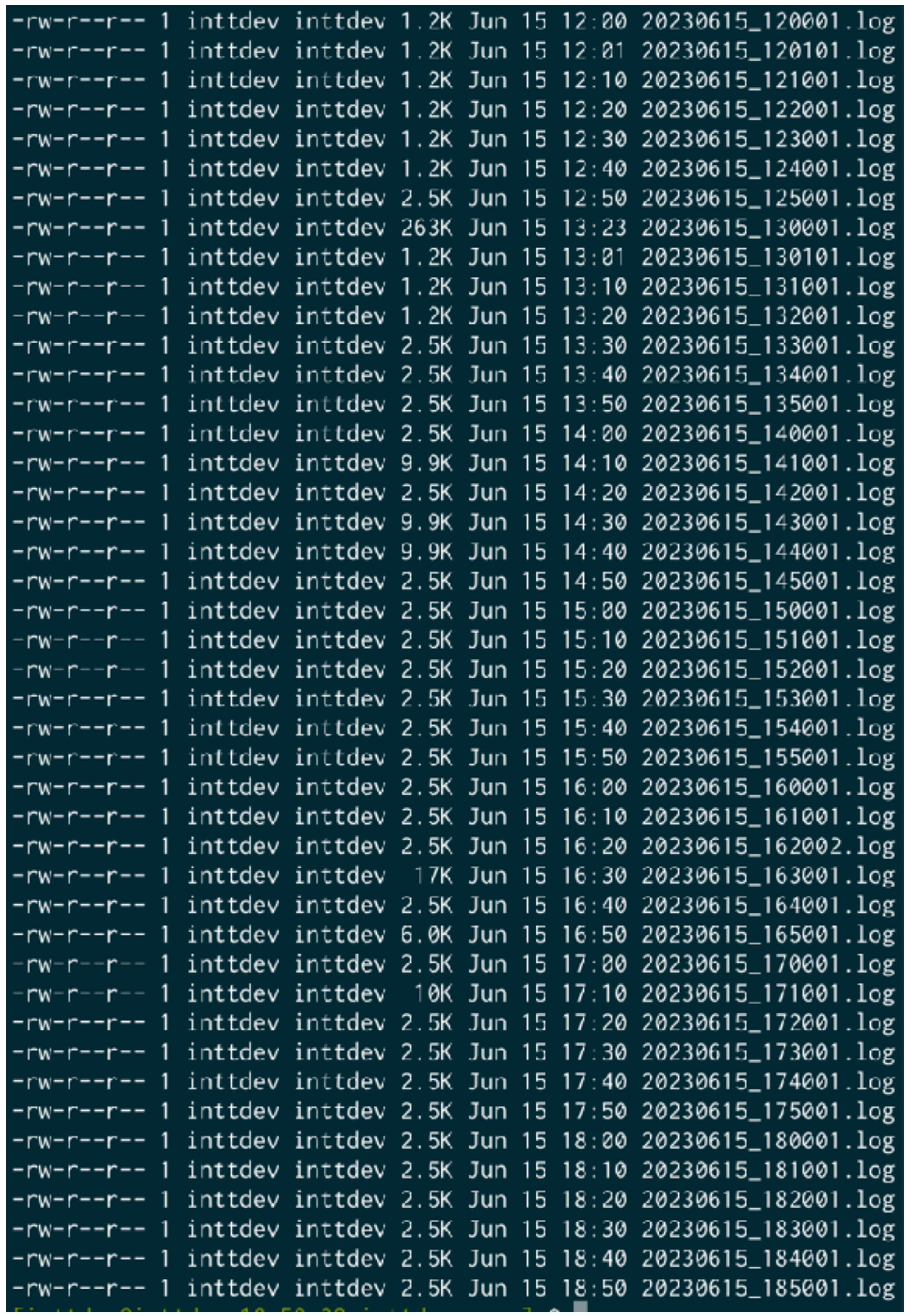
Logging

- Log files are dumped at inttdev@inttdaq:/home/inttdev/INTT/log/inttdaq_cron

Concern

- Exclusive operation works within a PC/server. If you run the script from your PC, it may conflict with the automated process in inttdaq. Confliction just consumes memory in intt[0-7]. In the worse case, it affects DAQ, but I haven't seen it.

Does it really work?



```
-rw-r--r-- 1 inttdev inttdev 1.2K Jun 15 12:00 20230615_120001.log
-rw-r--r-- 1 inttdev inttdev 1.2K Jun 15 12:01 20230615_120101.log
-rw-r--r-- 1 inttdev inttdev 1.2K Jun 15 12:10 20230615_121001.log
-rw-r--r-- 1 inttdev inttdev 1.2K Jun 15 12:20 20230615_122001.log
-rw-r--r-- 1 inttdev inttdev 1.2K Jun 15 12:30 20230615_123001.log
-rw-r--r-- 1 inttdev inttdev 1.2K Jun 15 12:40 20230615_124001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 12:50 20230615_125001.log
-rw-r--r-- 1 inttdev inttdev 263K Jun 15 13:23 20230615_130001.log
-rw-r--r-- 1 inttdev inttdev 1.2K Jun 15 13:01 20230615_130101.log
-rw-r--r-- 1 inttdev inttdev 1.2K Jun 15 13:10 20230615_131001.log
-rw-r--r-- 1 inttdev inttdev 1.2K Jun 15 13:20 20230615_132001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 13:30 20230615_133001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 13:40 20230615_134001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 13:50 20230615_135001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 14:00 20230615_140001.log
-rw-r--r-- 1 inttdev inttdev 9.9K Jun 15 14:10 20230615_141001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 14:20 20230615_142001.log
-rw-r--r-- 1 inttdev inttdev 9.9K Jun 15 14:30 20230615_143001.log
-rw-r--r-- 1 inttdev inttdev 9.9K Jun 15 14:40 20230615_144001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 14:50 20230615_145001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 15:00 20230615_150001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 15:10 20230615_151001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 15:20 20230615_152001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 15:30 20230615_153001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 15:40 20230615_154001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 15:50 20230615_155001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 16:00 20230615_160001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 16:10 20230615_161001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 16:20 20230615_162002.log
-rw-r--r-- 1 inttdev inttdev  17K Jun 15 16:30 20230615_163001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 16:40 20230615_164001.log
-rw-r--r-- 1 inttdev inttdev 6.0K Jun 15 16:50 20230615_165001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 17:00 20230615_170001.log
-rw-r--r-- 1 inttdev inttdev  10K Jun 15 17:10 20230615_171001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 17:20 20230615_172001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 17:30 20230615_173001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 17:40 20230615_174001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 17:50 20230615_175001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 18:00 20230615_180001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 18:10 20230615_181001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 18:20 20230615_182001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 18:30 20230615_183001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 18:40 20230615_184001.log
-rw-r--r-- 1 inttdev inttdev 2.5K Jun 15 18:50 20230615_185001.log
```

← 13:00, processing 11 runs by hand

← 14:10, Run00012003

← 14:30, Run00012005

← 14:40, Run00012008

← 16:30, Run00012020, 12021

← 16:50, Run00012023

← 17:10, Run00012030

time

Log files

List of runs in bbox:

- 00012003
- 00012004
- 00012005
- 00012006
- 00012008
- 00012009
- 00012010
- 00012011
- 00012020
- 00012021
- 00012022
- 00012023
- 00012025
- 00012030
- 00012031

Some runs were processed automatically. The skipping issue is due to an incomplete algorithm in my code.