Russell Folk
C S 521

# Pthreads Programming Report

This report details the work that went into writing this program package. It will be separated into the following categories: Record Format, Checksum Algorithm, Writer program, Checker program. The *writer* and *checker* programs and supporting libraries are written in C++ and are compiled using the supplied Makefile. The scripts that provide automated *device-file* generation and a complete experiment runner are written in bash and will run on any system that has bash installed.

## Record Format

As specified in the homework requirements, a record is 8kB in size and includes the following information:

- Thread ID: id assigned to the thread that created the record (starts at 0)
- Record number: number of records that have been created by this thread so far (starts at 1)
- Record address: address in the *device-file* that the record is supposed to be written
- Time stamp: number of milliseconds since the UNIX epoch that the record was created
- Random data: data that is generated based on previous data and fills the 8kB record
- Checksum: a checksum of the entire record save for the spots filled by the checksum (see Checksum Algorithm)

A record is an array of 1024 long values. The first and last values are reserved for the checksum, both spots are used for redundancy in the case of a partial write. The address is stored in the record so that the *checker* may validate against a missed-write, that is a write that is supposed to go to one location but ends up in another. Due to the implementation of C++ however, this never happens, or rather has not been observed and would be very interesting to determine how it happens if it did. The time stamp is stored for two reasons: first, to check whether the last record written by the thread is complete or partial and second, to be able to estimate the time of “power loss” inside the *checker* program.

Random data is created by adding the previous value in the array plus the checksum times the thread id times the array position plus the record number. This allows for the *checker* program to recreate a record and check how much of a partial record was written to the *device-file*. Further information can be read in the documentation.

## Checksum Algorithm

For a checksum algorithm, I implemented the Fletcher algorithm for 64-bit. Further information can be read in the documentation.

## Writer Program

The function of the *writer* program is to run a multi-threaded instance that creates and writes records to a *device-file* indefinitely. The biggest challenge with the *writer* was generating partial writes, it turns out that pwrite (required by the assignment specifications) is “atomic” in that the POSIX system will not interrupt a write. In order to better emulate a power loss, I write each long in the record sequentially so that an interrupt can hit at any point in the 1024 record pieces. Further information can be read in the documentation.

### Checker Program

The function of the *checker* program is to validate the *device-file* in a single thread. Using the information from the records, the *checker* is able to provide the total number of threads that wrote to the *device-file*, and the total number of records written by each thread. This is further separated into complete (successful) writes and incomplete writes. As an additional feature, the *checker* attempts to analyze the percentage of all record pieces written by totaling the number of pieces written and dividing by the total number of pieces that should have been written. Another feature added is checking for missed-writes, where a record was assigned to one address but written to another. As expected, I never saw an instance where this occurred, but this feature was tested to work in a forced miss-write. The final additional feature provided is an estimated time of “power loss” which works off of the time stamps stored in the record data. Further information can be read in the documentation.

## Obtaining and running the program

This simulation software may always be obtained on GitHub in the Pthread_Power_Fault_Tester repository located at https://github.com/russellfolk/Pthread_Power_Fault_Tester. To obtain the latest source code, run `git clone https://github.com/russellfolk/Pthread_Power_Fault_Tester.git` and refer to the README.md document.