Process Overseer

Criteria Sheet

Criteria sheet version: 1.00

Marks: 40

A: 10 marks

Check	Marks	Criterion
☐	1	Overseer and controller correctly use the provided IP address and port arguments and are able to establish a connection with one another.
☐	1	Controller terminates immediately. If the controller cannot establish a connection with the server it logs the connection failure message before terminating.
☐	1	Overseer executes indefinitely, handling subsequent client requests.
☐	1	Requested file is correctly executed.
☐	1	Requested file is correctly executed with an arbitrary number of arguments, where the first argument to the process is the file path provided to the overseer. All arguments are present in the logging statements.
☐	1	Status code is correctly reported when the process terminates.
☐	1	Execution is correctly performed with a fork and exec structure.
☐	1	Expected overseer logging under normal circumstances is present with the correct times (within an error margin of 2 seconds).
☐	1	Expected overseer logging when the file to be executed does not exist is present with the correct times (within an error margin of 2 seconds).
☐	1	Expected log time format is correct.

B: 4 marks

Check	Marks	Criterion
☐	1	Expected usage message is printed when `-help` is the first argument.
☐	1	Logging is correctly redirected to the path specified optionally by `-log`.
☐	1	Stdout and stderr of the child is correctly redirected to the path optionally specified by `-o`.
☐	1	Controller arguments are validated. The usage message must be printed before terminating if any of the following circumstances are met: an unknown argument is provided, argument order as per the usage specification is violated, or if a non-integer is provided as a port. *Note: argument validation for sections D and E is not required; the markers will only test valid arguments for these sections.*

C: 5 marks

Check	Marks	Criterion
☐	1	Thread pool is implemented correctly and all five threads are started at server initialization.
☐	1	Queue is implemented as a linked list and is shared by all threads (does not have to exist in global scope). Insertion (appending or prepending) into the linked list is correctly performed such that a FIFO queue is affected.
☐	1	Request handling is exclusively performed by one thread when signalled by a condition variable. Main thread is only responsible for client acceptance, request insertion, and condition variable signalling.
☐	1	Mutexes synchronize all access to the queue.
☐	1	Threads await new requests after handling a request.

D: 4 marks

Check	Marks	Criterion
☐	1	`SIGTERM` is sent to the child after 10 seconds.
☐	0.5	`-t seconds` correctly modifies the `SIGTERM` timeout to `seconds`.
☐	0.5	If the child is still alive 5 seconds after sending `SIGTERM`, it is killed via `SIGKILL`.
☐	1	`SIGINT` handling is implemented by killing all children and freeing all dynamic memory (Valgrind must report zero memory leakage).
☐	1	Expected logging is present with the correct times (within an error margin of 2 seconds).

E: 8 marks

Check	Marks	Criterion
☐	1.5	Maps file is correctly parsed to produce the correct byte values; only entries with an inode of 0 are used.
☐	1	Memory history is implemented as a linked list and is shared by all threads (does not have to exist in global scope). Memory usage is recorded every second. Without section C, this can be implemented as a global clock, otherwise if section C is implemented this is necessarily at the thread-level where each thread has its own clock.
☐	1	`mem` report contains the correct values with the specified format.
☐	1	`mem <pid>` report contains the correct values (times are within an error margin of 2 seconds) with the specified format.
☐	1.5	`memkill` works correctly and `sysinfo` was used to determine the total usable main memory size for memory consumption calculations.
☐	1	`mem` reports are written to controller's stdout (not the overseer).
☐	1	Everything is handled by a thread and all race conditions are avoided with mutexes. That is, `mem [pid]` and `memkill` are treated as ordinary requests to be handled by threads, including the communication with the client. This criterion only applies if section C is implemented, otherwise this criterion is awarded by default if any of the other criteria in this section have been satisfied.

Makefile: 1 mark

Check	Marks	Criterion
☐	0.5	Running `make` compiles executables `overseer` and `controller`. No directories are created.
☐	0.5	Running `make clean` removes `overseer` and `controller`.

Implementation quality: 8 marks

Check	Marks	Criterion
☐	1	Zero runtime errors such as memory access violations and deadlocks.
☐	1	Appropriate memory management including zero or very little memory leakage over the duration of the overseer's execution.
☐	1	Files and sockets are closed when no longer in use.
☐	1	Appropriate network communication strategies are applied and network byte order is used for any non-character data.
☐	1	Busy-waiting is avoided and sleeping has a maximum resolution of one second so that the overseer and any of its threads are responsive to events such as child termination, new client requests, etc.
☐	1	Code is almost entirely non-redundant and KISS (keep it simple, stupid) is adhered to.
☐	1	Consistent application of standard conventions with respect to identifier naming and whitespace. Magic numbers are avoided via #define directives.
☐	1	Code is mostly commented but not excessively. Comments **concisely** state the purpose of source files, header files, data structures, functions, and non-trivial code blocks.