

Concurrent Supermarket Simulator

Processes, POSIX threads, Unix sockets, and dynamic checkout control

Author: Giuseppe Muschetta

University: University of Pisa, B.Sc. in Computer Science

Course A instructor: Prof. G. Prencipe

Assignment edition: academic year 2019/2020

Abstract

This C project models a supermarket in which customers and cashiers are threads within a supermarket process, while a separate director process decides when to open or close checkouts. Capacity, timing, and decision thresholds are configurable. Shopping, FIFO queues, customer migration, periodic reports, and two shutdown modes interact to produce an evolving workload.

The project combines process management, synchronization, local communication, and observability in one system. This report revisits the original document [1] and explains the supplied implementation [3], distinguishing its design from the validation evidence available for this review.

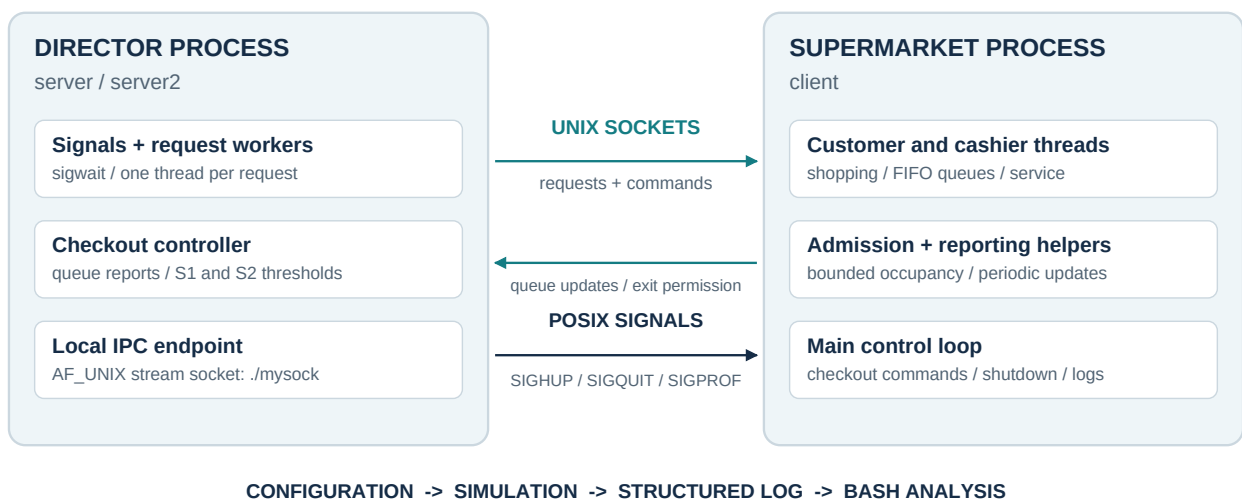


Figure 1. Architecture reconstructed from the source. Sockets carry requests and responses; signals coordinate changes of state.

1. Model and scope

The assignment [2] limits the system to **C customers** and **K checkouts**. A configured subset of checkouts opens initially. Each customer selects a shopping duration and a basket size; customers buying nothing request permission to leave from the director. Others join an open checkout and can reconsider their queue. The model also handles checkout closure and the relocation of waiting customers.

The scope is a local, two-process simulation for studying operating systems concepts. The executables are named **server**, **server2**, and **client**; the last represents the entire supermarket process. The project provides no quantitative throughput benchmark.

2. From startup to shutdown

Startup and initialization

The Makefile builds two director variants. **server** waits for a separately started supermarket; **server2**, compiled with **BUILD2**, uses **fork** and **exec** to start **./client**. The director creates **./mysock** and accepts an initial persistent connection. The processes exchange PIDs and the configuration pathname, then read their respective parameters.

The supermarket allocates its checkout array and statistics structures, opens the initial checkouts, and creates **C** customers. Admission waits until occupancy falls to **C-E** or below, then refills the supermarket to **C**. Additional departures before the refill can make the admitted group larger than **E**.

Concurrent roles

Component	Responsibility
Director: signal thread	Uses sigwait; forwards SIGHUP/SIGQUIT and acknowledges SIGPROF notifications.
Director: detached workers	One thread per accepted request: queue reports, exit permission, or final notification.
Supermarket: customers	Simulated shopping, checkout selection, optional queue migration, and exit records.
Supermarket: cashiers	FIFO service, statistics, and closure of their assigned checkout.
Support threads	Admission manager, reporting helper per checkout, and workers for director commands.

Opening and closing policy

The director keeps an estimate of each queue length; **-1** denotes a closed checkout. A queue with at least **S2** customers can trigger the opening of the first available checkout. At least **S1** checkouts with zero or one customer can trigger a closure, while keeping one checkout open. The scan stops at the first triggering criterion, so scan order affects the decision.

Two shutdown modes

Signal to the director	Supermarket behavior
SIGHUP	Stops admission and lets customers already inside complete their shopping and checkout service.
SIGQUIT	Starts evacuation: customers can abandon shopping or their queue. Termination remains cooperative.

Both paths wait for the threads covered by the shutdown protocol, notify the director, and produce the log. Emergency closing allows unfinished purchases to be abandoned; sleep calls and service already in progress can still finish.

3. Coordination and data structures

Sockets carry data; signals coordinate control

Communication uses `AF_UNIX/SOCK_STREAM`. The initial connection remains open for director commands. Checkout reports, requests to leave without purchases, and the final closure notification use dedicated connections. Integer message codes are defined in `shared.h` and converted using `htonl/ntohl`.

Exchange	Essential sequence
Initialization	Configuration-path length; PID exchange; request and delivery of the NUL-terminated pathname.
Checkout report	Update request; checkout identifier; number of queued customers; acknowledgment.
Exit without purchases	Exit request; basket-size check; permission granted when the product count is zero.
Open/close command	SIGPROF to the supermarket; SIGPROF reply; command and checkout index on the persistent socket; outcome.

The director signal thread uses a condition variable to wake the worker waiting for the SIGPROF acknowledgment. Signals are blocked in the threads and received synchronously through `sigwait`; SIGPIPE is ignored. The `readn` and `writen` wrappers retry partial operations and EINTR interruptions. Decoding an integer still requires an additional check for EOF before all its bytes have arrived.

Mutual exclusion and waiting conditions

The supermarket uses a global mutex when updating shared state and structures. Each checkout has a condition variable: a cashier with an empty queue sleeps until a customer arrives or closure is ordered. The admission condition variable notifies the entry manager when a customer leaves.

This design makes coordination explicit, with a significant performance consequence: the cashier holds the global mutex during the service `nanosleep (client.c:594-630)`. Service sections at different checkouts are therefore serialized. Multiple cashier threads alone provide no evidence of parallel service or increasing throughput as `K` grows.

FIFO queues and checkout migration

The generic queue is a singly linked list with head, tail, and length. `push` and `pop` take $O(1)$; position lookup and arbitrary removal take $O(n)$. The module contains no locks, leaving synchronization to its caller. A customer compares the people ahead with other queue lengths and rechecks the destination when moving.

Records of departed customers remain in memory until the log is written. Waiting-customer queues and timing queues have different payload ownership; deallocation must follow each payload lifetime.

4. Configuration and observability

The two supplied configuration files define different workloads. The parser uses **line positions** and the ":" delimiter, so ordering must be preserved. Durations in the table are in milliseconds unless otherwise stated.

Parameter / line	Meaning	config1	config2
K / 1	Maximum number of checkouts	2	6
C / 2	Maximum customers inside	20	50
E / 3	Departures threshold for admission refill	5	3
S / 4	Queue reconsideration interval	30	20
T / 5	Maximum shopping duration	500	200
P / 6	Maximum basket size	80	100
A / 7	Initially open checkouts	1	2
Log / 8	Statistics output filename	resoconto.log	resoconto2.log
d / 9	Service time per product	2	2
Interval / 10	Reporting period to the director	20	20
S1 / 11	Closing threshold	1	2
S2 / 12	Opening threshold	4	5

Modeled durations and observed time

A customer selects an integer shopping duration from 10 to T ms and a product count from 0 to P. At each opening, a checkout selects a fixed component b from 20 to 80 ms. Requested service is $t = b + d \times \text{product_count}$. Stay and opening durations are measured with `gettimeofday`; stored service durations come from the model.

From log records to a readable summary

The first two log lines contain total customers and purchased products. They are followed by six lines per departed customer and six per checkout, including all K slots. **analisi.sh** parses these blocks, sums opening periods, and calculates average service time using Bash, awk, and bc.

Entity	Information in the summary
Customer	ID, thread identifier, purchased products, total stay, checkout-phase time, and number of queues joined.
Checkout	ID, products processed, customers served, total open time, average modeled service, and closure count.

The summary presents durations in seconds with three decimal places. Checkout-phase time includes service and the delay until the customer observes completion; it does not isolate waiting before service.

5. Reproducibility and technical assessment

Intended execution environment: Linux

The Makefile is the build reference: C99, **-Wall -pedantic, -g -O3**, and pthread linkage. This example starts the director variant that creates the supermarket, then requests normal closing after startup:

```
make
./server2 config2.txt &
director_pid=$!
sleep 10
kill -HUP "$director_pid"
wait "$director_pid"
bash ./analisi.sh resoconto2.log
```

test1 uses Valgrind and sends SIGQUIT after 15 seconds; **test2** uses config2 and sends SIGHUP after 25 seconds. Both require GNU timeout; analysis requires Bash, awk, and bc. These targets create logs and other files, so preserving the original archive requires a working copy.

Check performed	Result and scope
Material review	Source, configurations, script, two PDFs, and ZIP archive inspected with the original project unchanged.
Isolated build	make completed with Apple clang 21 on macOS arm64; server, server2, and client built with warnings.
Script syntax	bash -n analisi.sh completed successfully.
Linux runtime tests	Not performed in this review; no claim of freedom from leaks, races, or runtime errors.

Engineering improvements

Static inspection identifies concrete areas for further work: **(1)** give detached workers a stable fd argument and close ordinary request descriptors (**server.c:306-353**); **(2)** require a complete integer before decoding (**shared.c:49-52**); **(3)** narrow the global lock and synchronize flags and counters consistently; **(4)** validate configuration and durations. Only tv_nsec is populated: delays of 1000 ms or more are represented incorrectly. The supplied settings stay below that limit.

UNIX_PATH_MAX=108 assumes the Linux layout. On macOS, sun_path is shorter and copying the pathname produces overflow warnings. A native build therefore does not establish portability. These observations come from source inspection and compiler warnings; simulation failures were not reproduced.

Documentation alignment and sources

The original report describes separate modules now combined in server.c and client.c and mentions SIGALRM; the supplied implementation uses SIGPROF. CMakeLists.txt is incomplete relative to the Makefile. The ZIP contains historical development caches and binaries, with no logs establishing test outcomes or quantitative performance.

[1] Relazione.pdf, 3 pages, original author report. [2] Traccia_progetto.pdf, Operating Systems Laboratory, assignment edition 2019/2020, revised 5 May 2020. [3] server.c/.h, client.c/.h, shared.c/.h, queue.c/.h, Makefile, analisi.sh, config1.txt, config2.txt, and cmake-build-debug.zip, supplied snapshot. Report revision and documentary review: 15 October 2026.