ITMO UNIVERSITY

National Research University ITMO

Faculty of Software Engineering and Computer Systems

SYSTEM SOFTWARE FUNDAMENTALS

Lab Work #5

Interprocess Communication and Thread Synchronization

Timothy Labushev

Group P3302

Saint Petersburg

2019

Assignment

Part I

Using C, implement a server that stores its `PID`, `UID`, `GID` on launch and continuously updates:

- The number of seconds passed since the process start;
- Average system load for the last 1, 5, and 15 minutes.

The server should support the following communication channels:

1. System V shared memory
2. System V message queue
3. Memory-mapped file

Part II

Write a progam in C which creates an array of lowercase Latin letters and mutates it in concurrently running threads. *Thread #1* inverts the case, *thread #2* reverses the array in place.

The following operations should be implemented:

1. Each second the main thread alternately wakes up one of the mutating threads, waits until they complete an operation, and outputs the array to `stdout`. *POSIX semaphores* are used for interthread synchronization.
2. Each second the main thread alternately wakes up one of the mutating threads, waits until they complete an operation, and outputs the array to `stdout`. *System V semaphores* are used for interthread synchronization.
3. The main thread outputs the array to `stdout` at a fixed time interval (in microseconds). The threads perform their operations at a fixed time interval (in microseconds) as well, locking the shared resource using *pthread mutexes*.
4. The main thread outputs the array to `stdout` at a fixed time interval (in microseconds). *Thread #1* and *thread #2* perform their operations at a fixed time interval (in microseconds) as well, acquiring the shared resource for writing. Yet another thread prints the number of uppercase letters in the array at a fixed time interval, acquiring the shared resource for reading. *pthread read-write locks* are used for interthread synchronization.

Part III

The following tasks should be done in C and Perl:

1. Implement a client-server pair communicating over Unix domain sockets. The behavior of the server should be based on the first part of the assignment.

2. Add custom signal handling for `HUP`, `INT`, `TERM`, `USR1`, `USR2` on the server.
3. Create a program that launches a child process via `fork()` and replaces its `stdin` with the read end of an unnamed pipe. The parent process outputs each second character of a file specified in arguments to the write end of the pipe. The child process launches the *wc* utility counting characters.

General Requirements

- Write a Makefile to compile the source code to separate executables, one for each assignment
- Insert `use strict; use warnings qw(FATAL all);` in Perl scripts and enable *taint mode* with `#!/usr/bin/perl -T`.

Code Listing

`.c` and `.pl` files are available at

`https://github.com/timlathy/itmo-third-year/tree/master/System-Programming-Fundamentals-5th-Term/Lab5-Interprocess-Communication`

Makefile

```
1  CC=gcc
2  CFLAGS=-std=c11 -g -Wall -Wextra -pedantic -Wno-address-of-packed-member
3
4  all: sysloadsrv sysloadclt \
5          semarray pthreadarray \
6          sysloadsocksrv sysloadsockclt unnamedpipe
7
8  sysloadsrv: sysloadsrv.c
9          $(CC) $(CFLAGS) -o sysloadsrv sysloadsrv.c
10
11 sysloadclt: sysloadclt.c
12         $(CC) $(CFLAGS) -o sysloadclt sysloadclt.c
13
14 semarray: semarray.c
15         $(CC) $(CFLAGS) -pthread -o semarray_posix -DSEM_POSIX semarray.c
16         $(CC) $(CFLAGS) -pthread -o semarray_sysv -DSEM_SYSV semarray.c
17
18 pthreadarray: pthreadarray.c
19         $(CC) $(CFLAGS) -pthread -o pthread_mutex -DMUTEX pthreadarray.c
20         $(CC) $(CFLAGS) -pthread -o pthread_rwlock -DRWLOCK pthreadarray.c
21
22 sysloadsocksrv: sysloadsocksrv.c
23         $(CC) $(CFLAGS) -o sysloadsocksrv sysloadsocksrv.c
24
25 sysloadsockclt: sysloadsockclt.c
26         $(CC) $(CFLAGS) -o sysloadsockclt sysloadsockclt.c
27
28 unnamedpipe: unnamedpipe.c
29         $(CC) $(CFLAGS) -o unnamedpipe unnamedpipe.c
30
31 clean:
32         @rm sysloadsrv sysloadclt \
33                 semarray_posix semarray_sysv \
34                 pthread_mutex pthread_rwlock \
35                 sysloadsocksrv sysloadsockclt \
36                 unnamedpipe
```