DISTRIBUTED TASK SCHEDULER

Oleksandr Razumov, Daulet Yerkinov

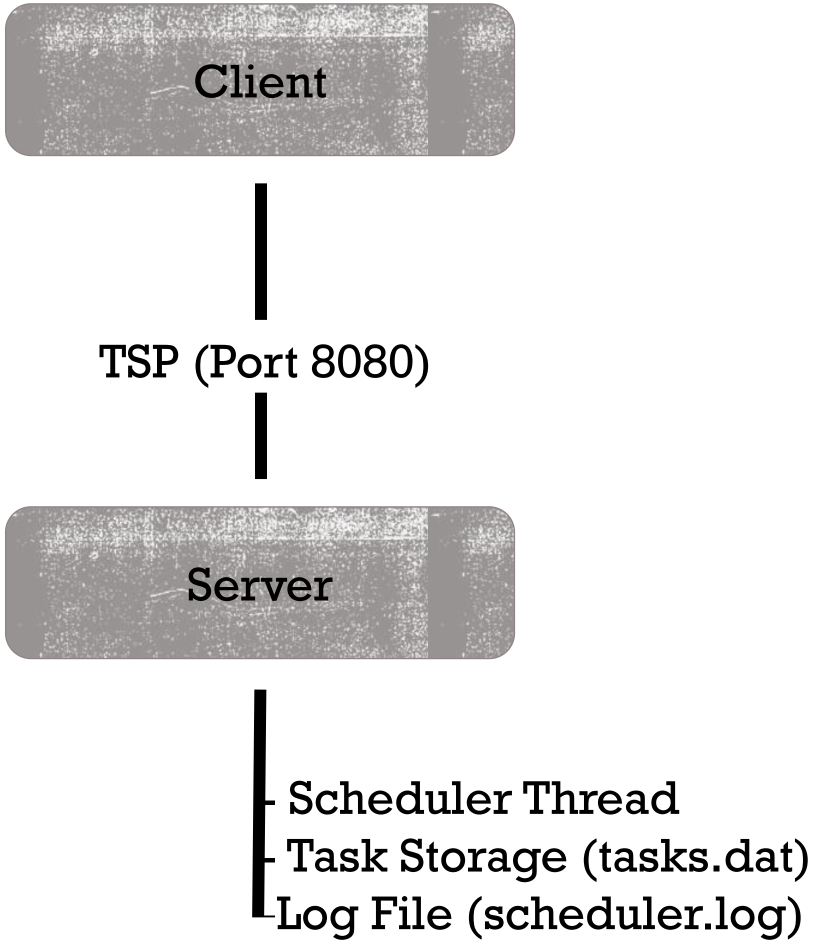
NScheduler

PROJECT OVERVIEW

- The Distributed Task Scheduler is a network-based system that allows users to:
- Connect to a remote server via TCP
- Authenticate securely
- Schedule commands for future execution
- Modify, delete, and inspect tasks
- Monitor execution status
- Store tasks persistently
- Log all system activity
- The system functions as a distributed version of **cron**, implemented fully in C++ using low-level OS mechanisms.

SYSTEM ARCHITECTURE

- **Server**
- TCP (Port 8080)
- Multi-threaded
- Background scheduler thread
- Task persistence (tasks.dat)
- Logging (scheduler.log)
- **Client**
- TCP connection
- Interactive menu
- Structured commands

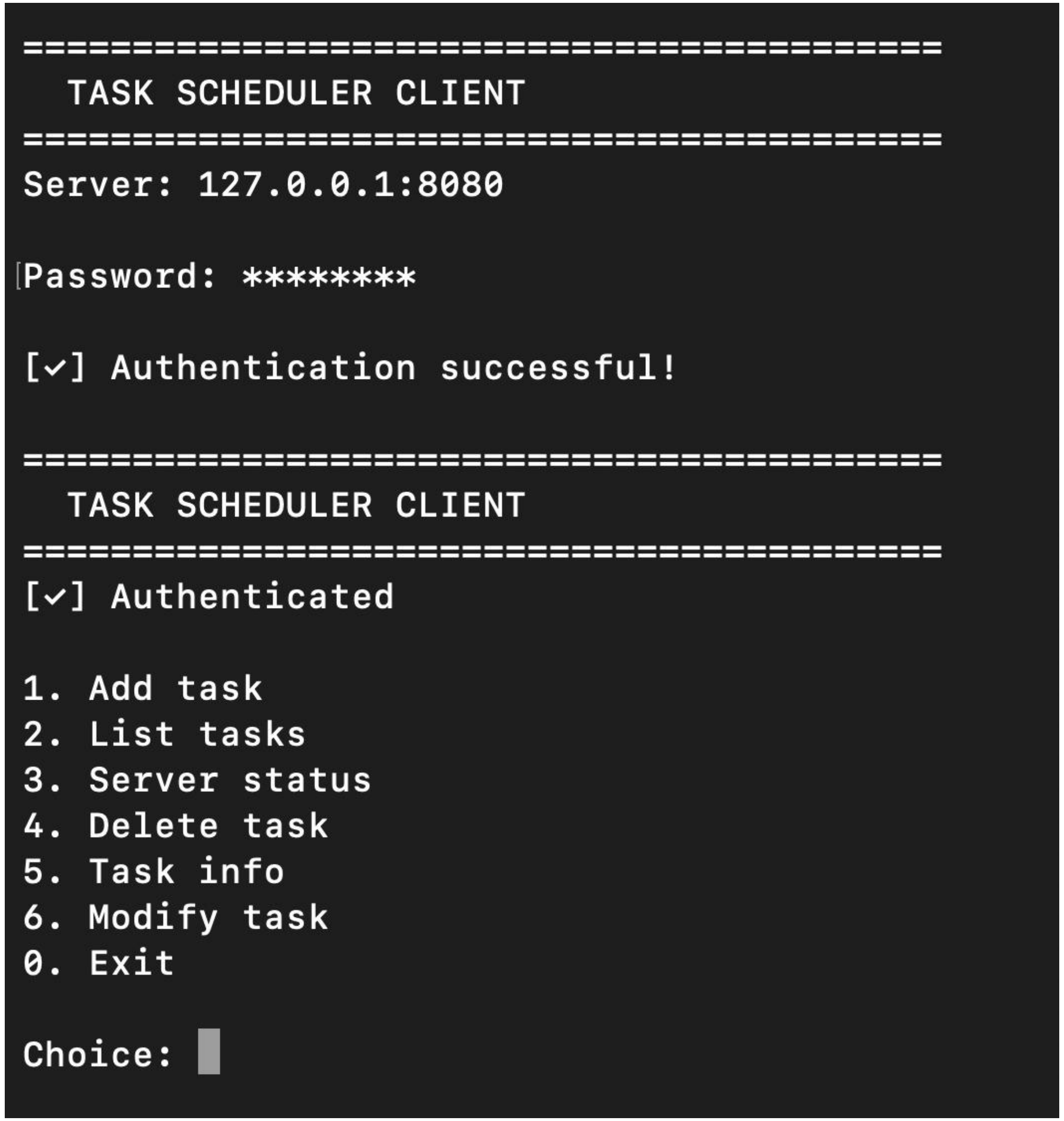


Client

TSP (Port 8080)

Server

Scheduler Thread
Task Storage (tasks.dat)
Log File (scheduler.log)



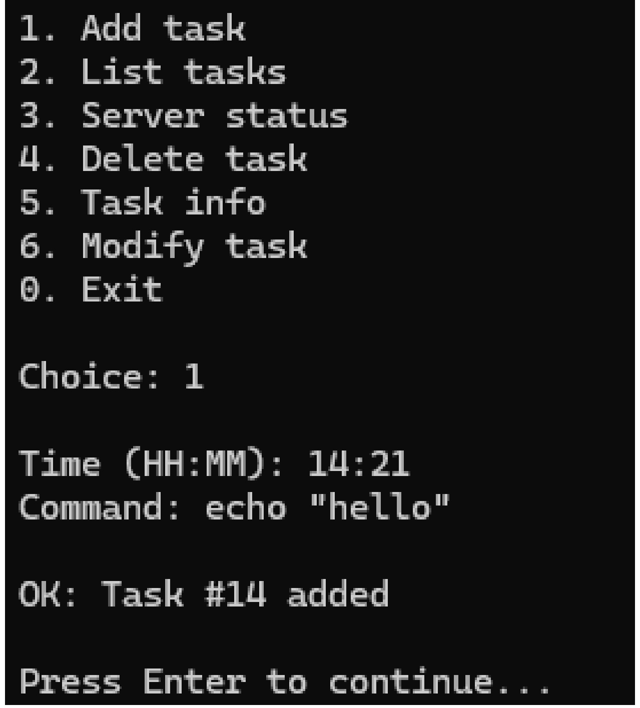
```
==========================================
  TASK SCHEDULER CLIENT
==========================================
Server: 127.0.0.1:8080

Password: ********

[✓] Authentication successful!

==========================================
  TASK SCHEDULER CLIENT
==========================================
[✓] Authenticated

1. Add task
2. List tasks
3. Server status
4. Delete task
5. Task info
6. Modify task
0. Exit

Choice: 
```

AUTHENTICATION & SECURITY

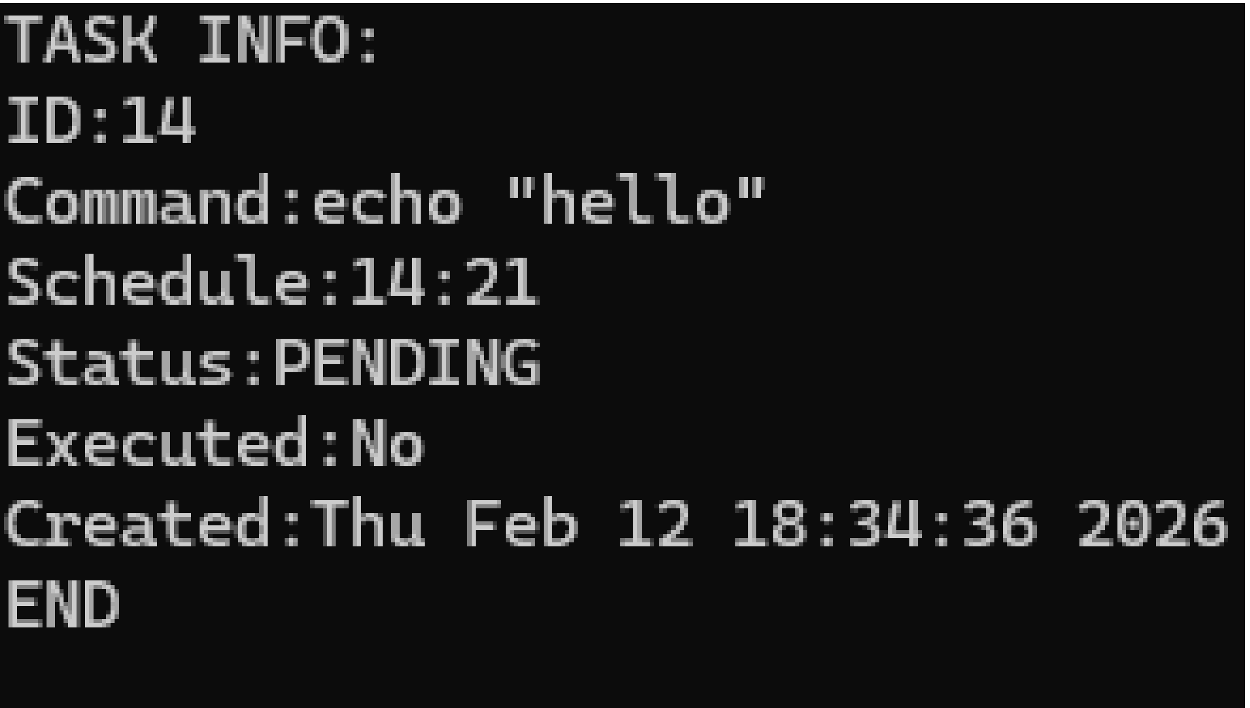
- Password required at startup
- Hidden input (termios, no echo)
- AUTH command sent to server
- Session stored by client IP
- Access denied if not authenticated

SCHEDULING LOGIC

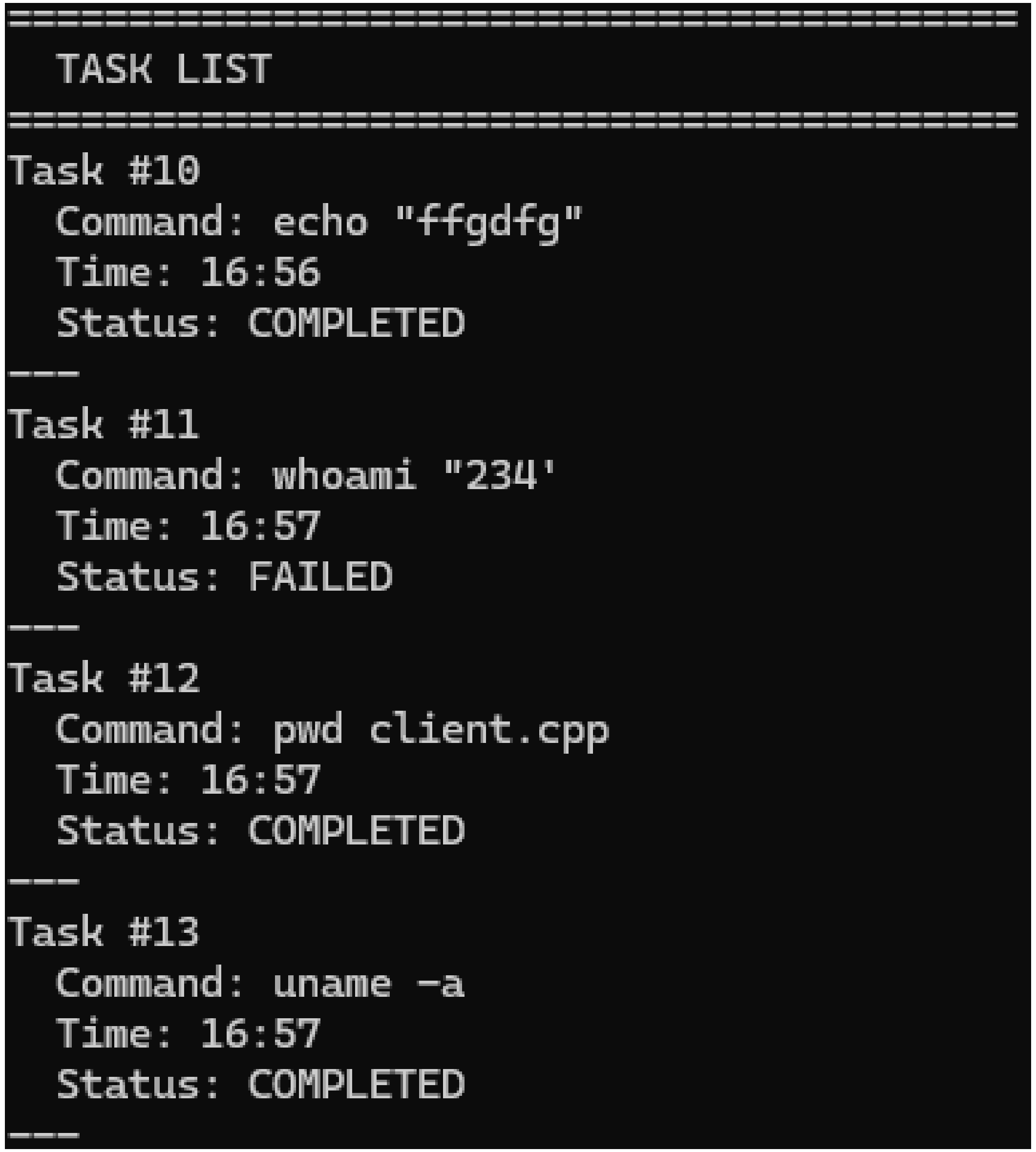
- Time format: HH:MM
- Converted to time_t
- Auto-shift to next day if needed
- Scheduler checks every 5 seconds
- Status updates automatically



```
1. Add task
2. List tasks
3. Server status
4. Delete task
5. Task info
6. Modify task
0. Exit

Choice: 1

Time (HH:MM): 14:21
Command: echo "hello"

OK: Task #14 added

Press Enter to continue...
```



```
TASK INFO:
ID:14
Command:echo "hello"
Schedule:14:21
Status:PENDING
Executed:No
Created:Thu Feb 12 18:34:36 2026
END
```

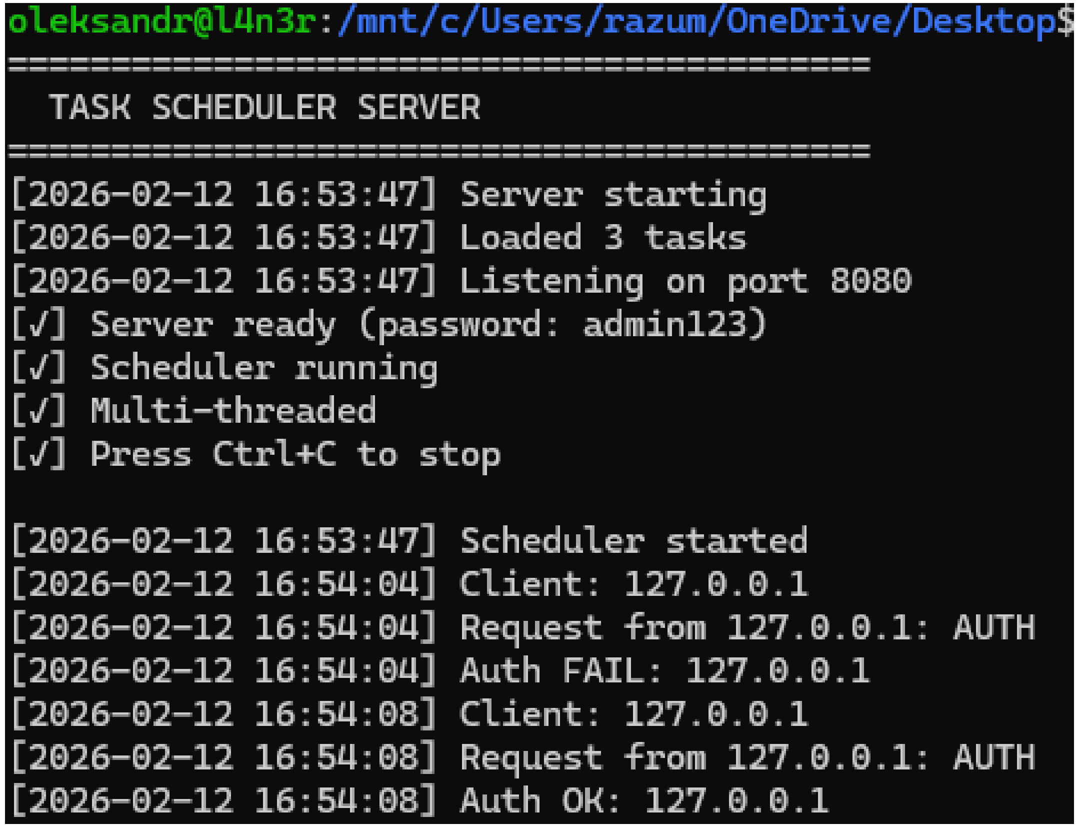


```
========================================
  TASK LIST
========================================
Task #10
  Command: echo "ffgdfg"
  Time: 16:56
  Status: COMPLETED
---
Task #11
  Command: whoami "234'
  Time: 16:57
  Status: FAILED
---
Task #12
  Command: pwd client.cpp
  Time: 16:57
  Status: COMPLETED
---
Task #13
  Command: uname -a
  Time: 16:57
  Status: COMPLETED
---
```

TASK EXECUTION

- Executed using system()
- Output redirected to log file
- Status:
- PENDING
- RUNNING
- COMPLETED
- FAILED

CONCURRENCY & SAFETY



```
oleksandr@l4n3r:/mnt/c/Users/razum/OneDrive/Desktop$
==========================================
  TASK SCHEDULER SERVER
==========================================
[2026-02-12 16:53:47] Server starting
[2026-02-12 16:53:47] Loaded 3 tasks
[2026-02-12 16:53:47] Listening on port 8080
[√] Server ready (password: admin123)
[√] Scheduler running
[√] Multi-threaded
[√] Press Ctrl+C to stop

[2026-02-12 16:53:47] Scheduler started
[2026-02-12 16:54:04] Client: 127.0.0.1
[2026-02-12 16:54:04] Request from 127.0.0.1: AUTH
[2026-02-12 16:54:04] Auth FAIL: 127.0.0.1
[2026-02-12 16:54:08] Client: 127.0.0.1
[2026-02-12 16:54:08] Request from 127.0.0.1: AUTH
[2026-02-12 16:54:08] Auth OK: 127.0.0.1
```

- Multi-threaded client handling
- Background scheduler thread
- Mutex protection:
- tasks
- sessions
- logs
- Signal handling (graceful shutdown)

OS CONCEPTS USED

Networking

- TCP socket programming
- Client–Server architecture
- Concurrent client handling

Concurrency & Synchronization

- Multi-threaded server
- Background scheduler thread
- Mutex protection (tasks, sessions, logs)
- Prevention of race conditions

Process & Execution Management

- System command execution (system())
- Output redirection
- Execution status tracking

Time & Signals

- Time-based scheduling (time_t, mktime)
- Periodic checking mechanism
- Signal handling (SIGINT, SIGTERM)
- Graceful shutdown with data persistence

File System

- Persistent task storage (tasks.dat)
- Log file management (scheduler.log)