Spyros Chatziargyros - it2024100

Operating Systems Lab - Log Analysis System

I built a system to monitor and analyze log files in UNIX. The project has 7 parts but I'll focus on the four most interesting challenges I encountered. I made this report using latex and I could not getting to work when using greek characters thats why it is in english

1 Reading Files the Hard Way (Part E)

The C program was probably the trickiest part. I needed to analyze log files using low-level system calls instead of nice functions like `fgets()`.

2 Making Things Run in Parallel (Part G)

My approach: Give each thread its own struct with separate memory. No shared state means no race conditions, which means I don't need mutexes. This keeps it simple - each thread does its job independently and writes results to its own space.

At first I was worried about thread safety, but then realized: if threads don't share data, there's nothing to protect. The main thread just waits for everyone to finish, then collects the results.

3 Tests

For testing i actually created a dedicated generator c program. It makes the following logic assumptions based on what the report said:

```
const char     *log_types[] =
  { "ERROR", "WARNING", "INFO", "CRITICAL", "FAILED" };
const char     *services[] =
  { "httpd", "sshd", "mysql", "nginx", "postgres" };
const char     *ips[] =
  { "192.168.1.", "10.0.0.", "172.16.0.", "203.0.113.", "198.51.100." };
```

After this simplifications we 5 log types, 5 sevices and 5 potential ip's. Based on these we take an argument of how many lines to include for tests.

I also added a 30-percent chance of getting a working or an error:

```
  // 30% chance of error/warning
  if (rand () % 100 < 30)
    {
      const char     *type = log_types[rand () % 5];
      const char     *service = services[rand () % 5];
      int             ip_last = rand () % 255;

      fprintf (fp, "%s: %s service issue at %s%d\n",
               type, service, ips[rand () % 5], ip_last);
    }
  else
    {
      fprintf (fp, "INFO: Normal operation of %s\n", services[rand () %
         5]);
    }
```

4 Regex and Pipes (Parts B & C)

Getting the right data out of logs required combining several UNIX tools.

The regex `^[0-9]{4}-[0-9]{2}-[0-9]{2}` matches dates at the start of lines. The dot in IP addresses needs escaping: `192\.168` because otherwise `.` matches any character.