OPERATING SYSTEMS & CONCURRENT PROGRAMMING IN ACTION

Part 10 of 11: Good Practices—Temporary Files & Logging

In this tenth session, we delve into two frequently overlooked yet essential aspects of systems programming: **managing temporary files** and **maintaining structured logs**. Neglecting either can lead to messy storage usage, debugging headaches, and security concerns. By referencing simple Bash scripts from https://github.com/antonioclim/OS5, we'll see "like for newbies" how to safely create and clean up temporary files and how to set up systematic logging and log rotation.

1. TEMPORARY FILE MANAGEMENT

1.1 Why It Matters

Temporary files are ubiquitous: compilers use them for intermediate object files, data pipelines store partial states, and applications buffer ephemeral data. **Risks** include:

- Accidental collisions (two processes creating the same temp file name).
- Storage bloat if not cleaned up.
- Security holes if sensitive data remains in an unprotected file.

1.2 Basic Pseudocode

```
create_temp_file():
  # 1) generate unique name
  # 2) open file safely
  # 3) return filename

do_stuff_with_temp_file(temp):
  # write intermediate data
  # read/modify as needed

cleanup_temp_file(temp):
  # remove file if it exists
```

1.3 Bash Script Demo: 8PCsystemResourcesV1.sh

Key Elements:

1. **Create** a secure, unique file name via mktemp.
2. **Use** that file for intermediate data.
3. **Remove** it at the end or on script exit.

Snippet:

```bash
#!/bin/bash

temp_file=$(mktemp /tmp/myapp.XXXXXX)
echo "Using temp file: $temp_file"

# store some data
echo "Intermediate data..." > "$temp_file"

# do stuff
sleep 5

# cleanup
rm -f "$temp_file"
echo "Temp file removed."
```

Notes:

- mktemp ensures uniqueness and sets correct permissions.
- The script might trap signals to ensure even if it's interrupted, the file gets removed (trap "rm -f $temp_file" EXIT).

---

2. STRUCTURED LOGGING

2.1 Importance

Logs document system behavior over time. Well-structured logs ease:

- **Troubleshooting** – pinpointing when and how something broke.
- **Performance Monitoring** – identifying patterns in CPU or memory usage.
- **Security Audits** – tracing suspicious activity.

However, logs can balloon in size. Automated cleanup (log rotation) is vital to avoid filling disks.

2.2 Simple Logging Pseudocode

```
every time interval:
  log_file = "log_" + current_timestamp + ".txt"
  write system metrics to log_file
  rotate_logs(max_to_keep=5)
```

2.3 Bash Script Demo: 8PCsystemResourcesV2REAL.sh

What It Does:

1. **Collects** CPU, memory, and disk usage.
2. **Appends** these metrics into a timestamped log file.
3. **Cleans** old logs beyond a retention threshold, e.g., only keep last 5 logs.

Snippet:

```
#!/bin/bash

log_dir="$HOME/logs"
mkdir -p "$log_dir"

timestamp=$(date +%Y%m%d_%H%M%S)
log_file="$log_dir/sys_usage_$timestamp.log"

echo "Timestamp: $(date)" > "$log_file"
echo "CPU:" >> "$log_file"
uptime >> "$log_file"

echo "Memory:" >> "$log_file"
free -h >> "$log_file"

echo "Disk:" >> "$log_file"
df -h >> "$log_file"

# rotate logs (keep 5)
ls -tp "$log_dir" | grep -v '/$' | tail -n +6 | xargs -I {} rm -- "$log_dir/{}"
```

Notes:

- **Timestamp** naming makes each log unique.
- A simple ls ... | tail -n +6 effectively discards all but the newest 5 logs.

3. WHY THESE PRACTICES MATTER

- **Safe Temp Handling**:
  - Minimizes collisions and security exposures.
  - Ensures leftover files don't clutter or fill storage.
- **Structured Logging**:
  - Speeds up debugging, analysis, and performance checks.
  - Automated cleanup prevents indefinite disk growth.

Together, they improve reliability and maintainability of any concurrent or multi-process environment.

---

4. CONCLUDING NOTES

Temporary Files: Always generate them uniquely, store only ephemeral data, and remove them promptly.

Logging: Capture relevant system data in a structured, time-stamped manner, then rotate or purge older logs to keep disk usage stable.

These "good practices" are as crucial for small, single-user systems as they are for multi-user, large-scale environments, ensuring consistent resource usage and simpler system administration.

REFERENCES

- Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts* (10th ed.). Wiley.
- Bash scripts available at https://github.com/antonioclim/OS5.
- *mktemp* and *trap* for secure temporary files: see man mktemp.

End of Part 10 of 11