Experiment 12: Building a Rule-Based Expert System

Name : Aditya Mishra

SAPid : 590029219

1. Theory

A Rule-Based Expert System uses predefined IF–THEN rules to make decisions. It imitates the reasoning process of a human expert.

Components:

- **Knowledge Base** – Stores rules.
- **Inference Engine** – Applies rules and makes decisions.
- **Working Memory** – Stores current facts.
- **User Interface** – Allows user inputs and displays results.

---

2. Process Automation and Job Scheduling

Cron – Repetitive Task Scheduler

- Edit cron jobs:
  `crontab -e`
- Syntax:
  `minute hour day month dayofweek command`
- Example:
  `0 9 * * * /home/user/backup.sh` (Runs daily at 9 AM)

at – One-Time Scheduler

- Example:
  `echo "backup.sh" | at 2:00 AM tomorrow`

---

3. System Administration Scripts

Automation tasks commonly include:

- Backup scripts
- User creation
- Log monitoring
- System health checks

Example rule:
IF disk_usage > 90% THEN alert "Disk Almost Full"

---

4. Services and Daemons (systemctl)

Common commands:

- `systemctl start service`
- `systemctl stop service`
- `systemctl status service`
- `systemctl enable service`
- `systemctl disable service`

---

5. Example Python Code for an Expert System

```
# Simple Rule-Based Expert System

facts = {}

rules = [
    {
        "conditions": {"temperature": lambda t: t > 80},
        "action": lambda: print("Alert: System Overheating!")
    },
    {
        "conditions": {"disk_usage": lambda d: d > 90},
        "action": lambda: print("Warning: Disk Almost Full!")
    }
]

def evaluate_rules():
    for rule in rules:
        if all(cond(facts.get(key)) for key, cond in
rule["conditions"].items()):
            rule["action"]()

# Example usage
facts["temperature"] = 85
facts["disk_usage"] = 75
evaluate_rules()
```