CRON Command Reference Guide

Overview

Cron is a time-based job scheduler in Unix-like operating systems. It enables users to schedule commands or scripts to run automatically at specified times, dates, or intervals.

Basic Concepts

- **Cron daemon (crond)** - Background service that runs scheduled tasks
- **Crontab** - Configuration file containing cron jobs
- **Cron job** - A scheduled task or command
- **Cron expression** - Time specification for when job should run

Crontab Format

```
# ┌──────────────── minute (0 - 59)
# │ ┌──────────────── hour (0 - 23)
# │ │ ┌──────────────── day of month (1 - 31)
# │ │ │ ┌──────────────── month (1 - 12)
# │ │ │ │ ┌──────────────── day of week (0 - 6) (Sunday to Saturday)
# │ │ │ │ │
# * * * * * command to execute
```

Cron Time Fields

1. **Minute (0-59)** - When during the hour to run
2. **Hour (0-23)** - Which hour of the day (24-hour format)
3. **Day of Month (1-31)** - Which day of the month
4. **Month (1-12)** - Which month of the year
5. **Day of Week (0-6)** - Which day of the week (0 = Sunday)

Special Characters

- `*` - Matches any value (wildcard)
- `,` - Separates multiple values
- `-` - Defines a range of values
- `/` - Defines step values

- ? - No specific value (alternative to *)
- L - Last (day of month or week)
- W - Weekday (closest weekday to specified day)
- # - Nth occurrence of weekday in month

Basic Commands

Managing Crontab

```bash
# View current user's crontab
crontab -l

# Edit current user's crontab
crontab -e

# Remove current user's crontab
crontab -r

# Install crontab from file
crontab filename

# View another user's crontab (as root)
crontab -u username -l

# Edit another user's crontab (as root)
crontab -u username -e
```

Cron Service Management

```bash
```

```bash
# Start cron service
sudo systemctl start cron    # Debian/Ubuntu
sudo systemctl start crond   # RHEL/CentOS

# Stop cron service
sudo systemctl stop cron

# Restart cron service
sudo systemctl restart cron

# Check cron service status
sudo systemctl status cron

# Enable cron to start at boot
sudo systemctl enable cron
```

Common Cron Expressions

Basic Timing Examples

```bash
# Run every minute
* * * * * /path/to/command

# Run at 5:30 AM every day
30 5 * * * /path/to/command

# Run at 2:00 PM every weekday
0 14 * * 1-5 /path/to/command

# Run every Sunday at midnight
0 0 * * 0 /path/to/command

# Run on the 1st of every month at 6:00 AM
0 6 1 * * /path/to/command
```

Using Special Characters

```bash
```

```
# Run every 5 minutes
*/5 * * * * /path/to/command

# Run every 2 hours
0 */2 * * * /path/to/command

# Run at 9:00 AM and 5:00 PM every day
0 9,17 * * * /path/to/command

# Run every weekday at 8:30 AM
30 8 * * 1-5 /path/to/command

# Run every 15 minutes during business hours
*/15 9-17 * * 1-5 /path/to/command
```

Complex Scheduling

```bash
# Run every quarter hour except at the top of the hour
15,30,45 * * * * /path/to/command

# Run every day except weekends
0 9 * * 1-5 /path/to/command

# Run on the first Monday of every month
0 9 1-7 * 1 /path/to/command

# Run twice a day (6 AM and 6 PM)
0 6,18 * * * /path/to/command

# Run every 6 hours
0 */6 * * * /path/to/command
```

Special Cron Strings

Instead of the five-field format, you can use these shortcuts:

```bash
```

```
@reboot    # Run once at startup
@yearly    # Run once a year (0 0 1 1 *)
@annually  # Same as @yearly
@monthly   # Run once a month (0 0 1 * *)
@weekly    # Run once a week (0 0 * * 0)
@daily     # Run once a day (0 0 * * *)
@midnight  # Same as @daily
@hourly    # Run once an hour (0 * * * *)
```

Examples with Special Strings

```bash
# Run backup script at reboot
@reboot /home/user/scripts/backup.sh

# Run monthly report
@monthly /home/user/scripts/monthly_report.sh

# Clean logs daily
@daily /usr/local/bin/cleanup_logs.sh

# Check disk space hourly
@hourly /home/user/scripts/check_disk.sh
```

Practical Examples

System Administration

```bash
```

```
# System backup every night at 2 AM
0 2 * * * /usr/local/bin/backup.sh > /var/log/backup.log 2>&1

# Update system packages weekly
0 3 * * 0 apt update && apt upgrade -y

# Clean temporary files daily
0 1 * * * find /tmp -type f -mtime +7 -delete

# Restart web server weekly
0 4 * * 0 systemctl restart apache2

# Monitor disk usage every hour
0 * * * * df -h | mail -s "Disk Usage Report" admin@example.com
```

Log Management

```bash
# Rotate logs daily at midnight
0 0 * * * /usr/sbin/logrotate /etc/logrotate.conf

# Compress old logs weekly
0 2 * * 0 gzip /var/log/*.log.1

# Clean old log files monthly
0 3 1 * * find /var/log -name "*.log" -mtime +30 -delete

# Send log summary daily
0 6 * * * tail -100 /var/log/syslog | mail -s "Daily Log Summary" admin@domain.com
```

Database Maintenance

```bash
```

```
# MySQL backup daily at 3 AM
0 3 * * * mysqldump -u root -ppassword database > /backup/db_$(date +\%Y\%m\%d).sql

# PostgreSQL vacuum weekly
0 2 * * 0 psql -d mydb -c "VACUUM ANALYZE;"

# Database health check hourly
0 * * * * /home/dba/scripts/db_health_check.sh

# Archive old records monthly
0 1 1 * * /home/dba/scripts/archive_old_records.sh
```

Web Development

```bash
# Deploy website daily at 2 AM
0 2 * * * cd /var/www/html && git pull origin main

# Clear cache every 6 hours
0 */6 * * * rm -rf /var/cache/app/*

# Generate sitemap weekly
0 1 * * 0 /home/user/scripts/generate_sitemap.php

# Check broken links monthly
0 4 1 * * /home/user/scripts/check_links.sh | mail -s "Broken Links Report" webmaster@example.com

# Update SSL certificates (Let's Encrypt)
0 12 * * * /usr/bin/certbot renew --quiet
```

Environment and Variables

Setting Environment Variables

```bash
```

```
# Set environment variables at the top of crontab
SHELL=/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=admin@example.com
HOME=/home/username

# Job using environment variables
0 5 * * * $HOME/scripts/backup.sh
```

Common Environment Issues

```bash
# Explicitly set PATH for commands
0 2 * * * /usr/bin/python3 /home/user/script.py

# Source profile for environment
0 3 * * * /bin/bash -c "source ~/.bashrc && /home/user/script.sh"

# Use full paths for reliability
0 4 * * * /usr/bin/find /home/user -name "*.tmp" -delete
```

Output and Logging

Redirecting Output

```bash
# Discard all output
0 2 * * * /path/to/command > /dev/null 2>&1

# Log output to file
0 2 * * * /path/to/command >> /var/log/cronjob.log 2>&1

# Email output (if MAILTO is set)
0 2 * * * /path/to/command

# Separate stdout and stderr
0 2 * * * /path/to/command >> /var/log/job.out 2>> /var/log/job.err

# Log with timestamp
0 2 * * * echo "$(date): Starting backup" >> /var/log/backup.log; /path/to/backup.sh >> /var/log/backup.log 2>&1
```

Email Configuration

```bash
# Set email recipient in crontab
MAILTO=user@example.com

# Multiple recipients
MAILTO=user1@example.com,user2@example.com

# Disable email
MAILTO=""

# Custom email subject and content
0 2 * * * /path/to/command || echo "Backup failed at $(date)" | mail -s "Backup Failure" admin@example.com
```

Security Considerations

File Permissions

```bash
# Crontab files should be properly secured
chmod 600 /var/spool/cron/crontabs/username

# Scripts should have appropriate permissions
chmod 700 /home/user/scripts/
chmod 755 /home/user/scripts/backup.sh
```

Best Practices

```bash
```

```
# Use full paths for security
0 2 * * * /usr/bin/rsync /source/ /destination/

# Validate input in scripts
#!/bin/bash
if [[ ! -d "/backup/destination" ]]; then
  logger "Backup destination not found"
  exit 1
fi

# Set restrictive umask
umask 077
0 3 * * * /home/user/scripts/sensitive_task.sh
```

Troubleshooting

Common Issues and Solutions

```bash
# Check if cron daemon is running
ps aux | grep cron
systemctl status cron

# Check cron logs
tail -f /var/log/cron
tail -f /var/log/syslog | grep cron

# Verify crontab syntax
crontab -l | crontab -

# Test command manually
sudo -u username /bin/bash -c "command"

# Check file permissions
ls -la /var/spool/cron/crontabs/
```

Debugging Cron Jobs

```bash
```

```
# Add debug output to jobs
0 2 * * * echo "Starting job at $(date)" >> /tmp/debug.log; /path/to/command >> /tmp/debug.log 2>&1; echo "Job finish

# Run job manually to test
/bin/bash -c "source /etc/profile; /path/to/command"

# Check command exists and is executable
which command
ls -la /path/to/command

# Verify environment variables
0 * * * * env > /tmp/cron_env.txt
```

Log Analysis

```bash
# Search for specific job in logs
grep "your_command" /var/log/cron

# Check for failed jobs
grep "FAILED\|ERROR" /var/log/cron

# Monitor cron activity in real-time
tail -f /var/log/cron

# Check job execution times
grep -E "$(date +%b\ %d)" /var/log/cron
```

Advanced Features

System-wide Cron Jobs

```bash
```

```
# /etc/crontab format (includes user field)
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *   * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6   * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6   * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6   1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
```

Cron Directories

```bash
# System cron directories
/etc/cron.hourly/   # Scripts run every hour
/etc/cron.daily/    # Scripts run daily
/etc/cron.weekly/   # Scripts run weekly
/etc/cron.monthly/  # Scripts run monthly

# Place executable scripts in these directories
sudo cp backup_script.sh /etc/cron.daily/
sudo chmod +x /etc/cron.daily/backup_script.sh
```

Anacron Integration

```bash
# /etc/anacrontab - runs missed jobs when system is up
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
RANDOM_DELAY=45

# period   delay   job-identifier   command
1       5     cron.daily      nice run-parts /etc/cron.daily
7       25    cron.weekly     nice run-parts /etc/cron.weekly
@monthly  45    cron.monthly    nice run-parts /etc/cron.monthly
```

Monitoring and Maintenance

Health Checks

```bash
# Check cron job execution
#!/bin/bash
# healthcheck.sh
if [ -f /tmp/backup_success ]; then
  echo "Backup completed successfully"
  rm /tmp/backup_success
else
  echo "Backup may have failed" | mail -s "Backup Alert" admin@example.com
fi

# Add to successful backup script
echo "success" > /tmp/backup_success
```

Performance Monitoring

```bash
# Monitor resource usage
0 * * * * (echo "$(date)"; ps aux | grep backup_script; free -h; df -h) >> /var/log/cron_performance.log

# Limit job runtime
timeout 3600 /path/to/long_running_script.sh

# Prevent overlapping jobs
#!/bin/bash
LOCKFILE=/tmp/backup.lock
if [ -f "$LOCKFILE" ]; then
  echo "Backup already running"
  exit 1
fi
touch "$LOCKFILE"
trap 'rm -f "$LOCKFILE"; exit' INT TERM EXIT
# Your backup commands here
```

Tips and Best Practices

1. **Always use absolute paths** for commands and files

2. **Test scripts manually** before adding to cron
3. **Set appropriate environment variables** at the top of crontab
4. **Use proper output redirection** to manage logs
5. **Implement proper error handling** in scripts
6. **Use lock files** to prevent overlapping jobs
7. **Monitor job execution** and set up alerts for failures
8. **Keep cron jobs simple** and delegate complex logic to scripts
9. **Use meaningful comments** in crontab files
10. **Regularly review and clean up** unused cron jobs

Common Pitfalls

- Using relative paths in cron jobs
- Not setting proper environment variables
- Assuming interactive shell features are available
- Not handling errors and failures
- Creating overlapping or conflicting jobs
- Not redirecting output properly
- Using unsupported cron syntax variations
- Not considering timezone issues
- Forgetting about daylight saving time changes
- Not testing jobs under cron environment conditions

See Also

- `at` - Schedule one-time jobs
- `systemd timers` - Modern alternative to cron
- `anacron` - Run missed periodic jobs
- `fcron` - Feature-rich cron implementation
- `systemctl` - Control systemd services and timers