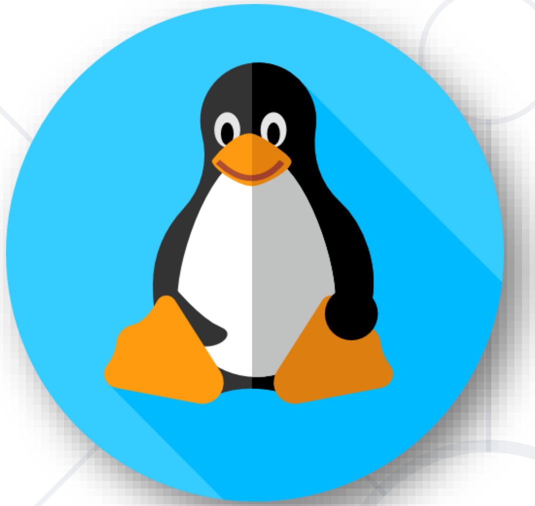
Bash Scripting. Automation

Bash Scripting Building Blocks

Repetitive Tasks Automation



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

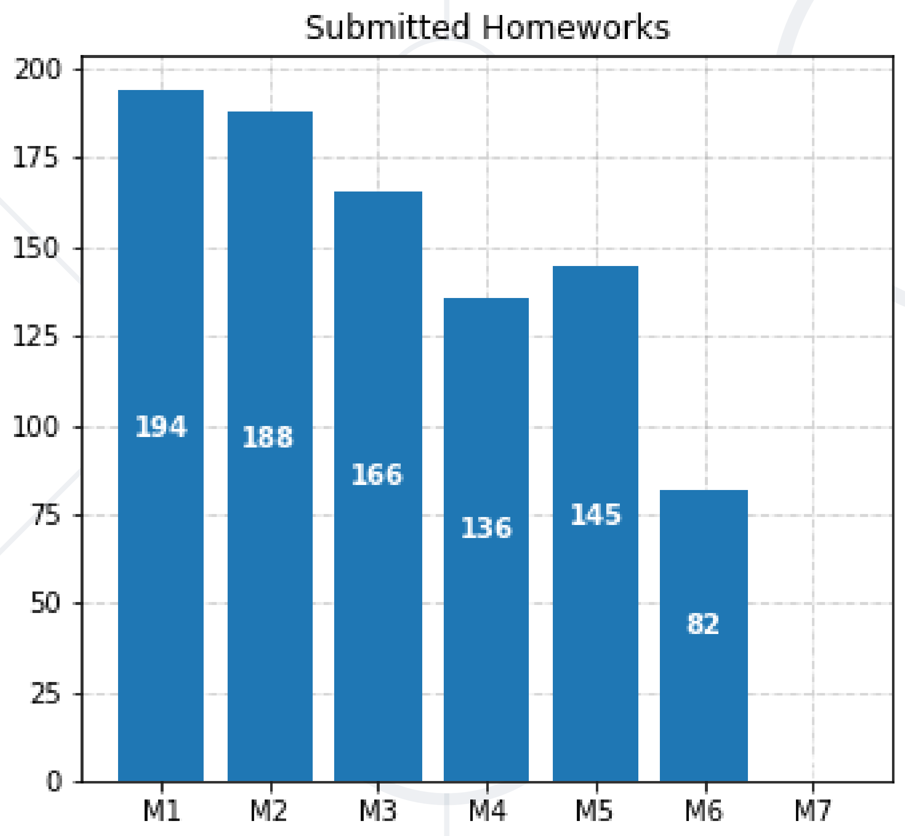
Have a Question?

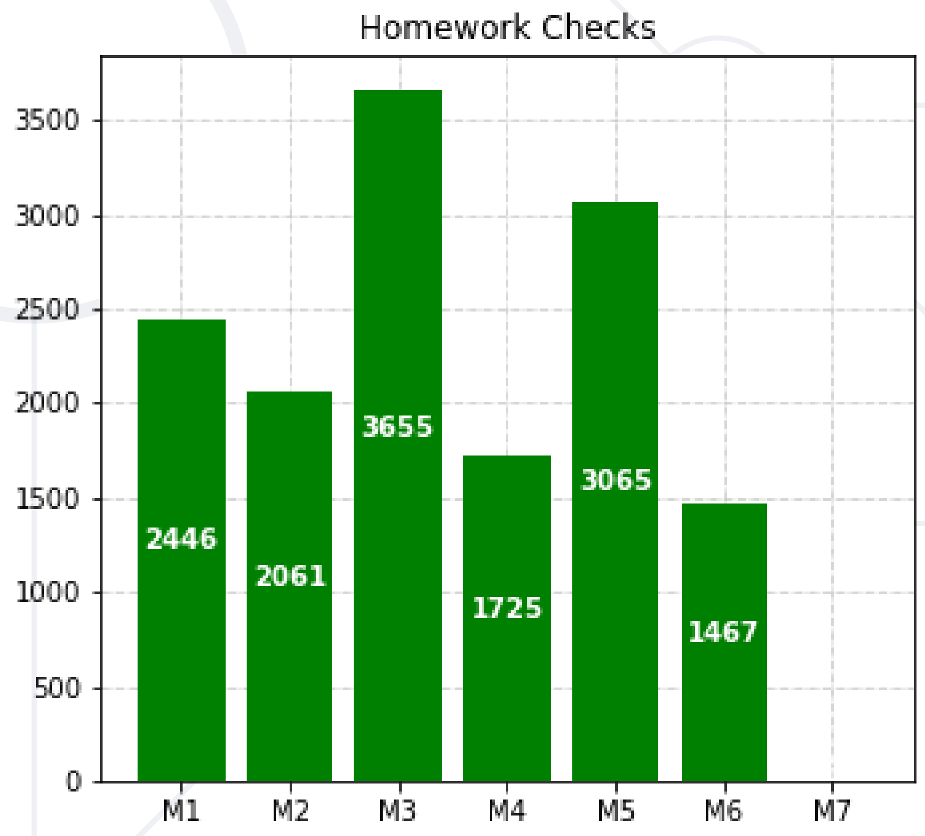
Software University

[sli.do](sli.do)

#LSA

Homework Progress





Solutions for **M6** can be submitted until **23:59:59** on **17.04.2025**

Solutions for **M7** can be submitted until **23:59:59** on **24.04.2025**

The End is Near ☺

THIS MODULE

AND ONE MORE TO GO

Exam is Coming, Prepare Yourself

Test Your Knowledge *

https://zahariev.pro/q/lsa

** It is hosted externally, and it is not part of SoftUni's infrastructure. Requires registration (sign up)*

Book Your Exam

By The End of This Week*

Check Your Profile

at SoftUni Web Site

There Should be an Exam Sign-up Form

** It could appear even earlier. There will be a message in the Facebook group when the sign-up form is available*

Previous Module (M6)

Quick Overview

What We Covered

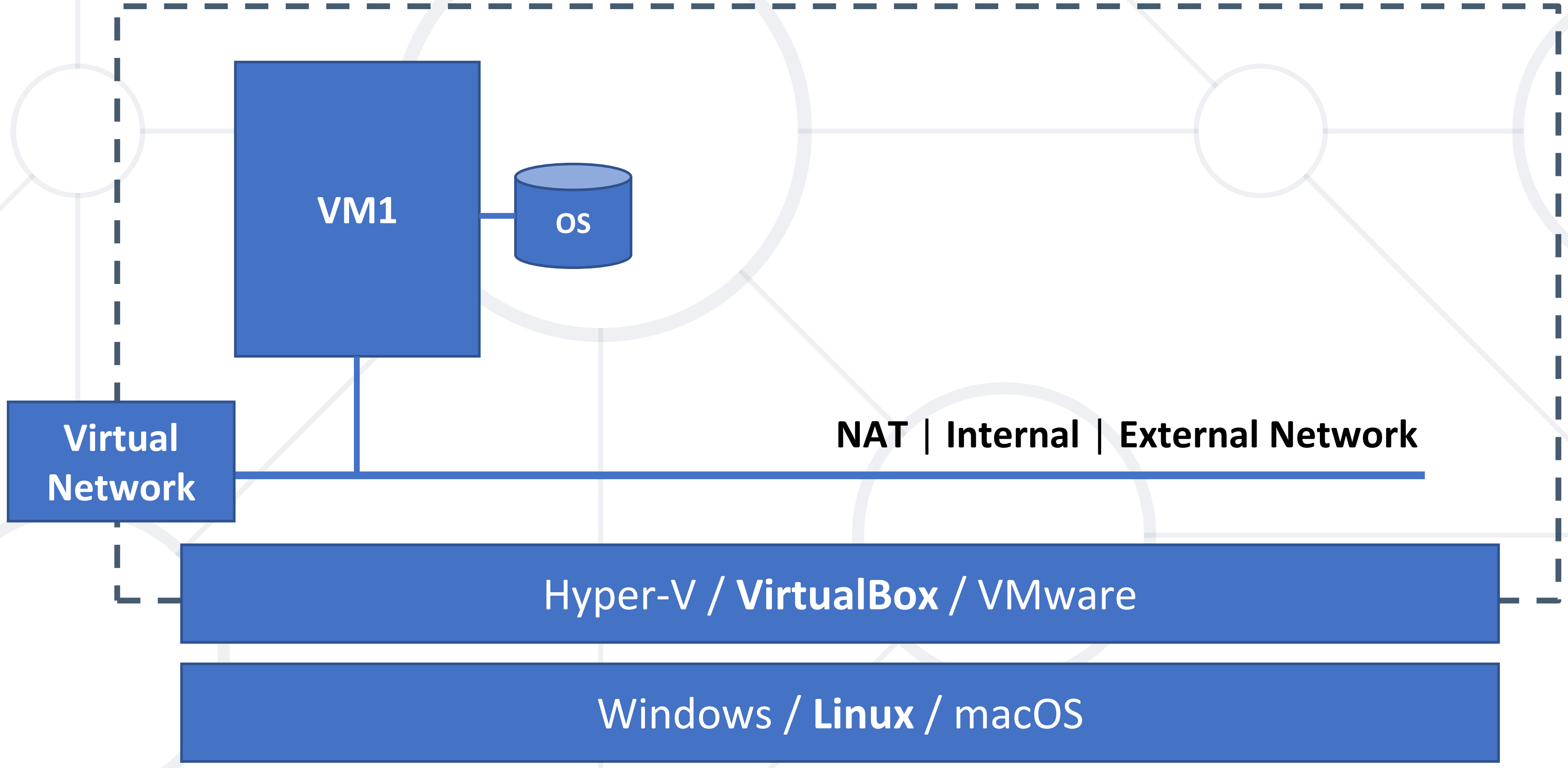
- Filesystem Hierarchy Standard (FHS)
- Archiving Tools
- Disks and Partitions Schemes
- File Systems

This Module (M7)

Topics and Lab Infrastructure

Table of Contents

Lab Infrastructure


VM1
OS
Virtual Network
NAT | Internal | External Network
Hyper-V / **VirtualBox** / VMware
Windows / **Linux** / macOS

Scheduling

Periodical Task Execution

Purposes

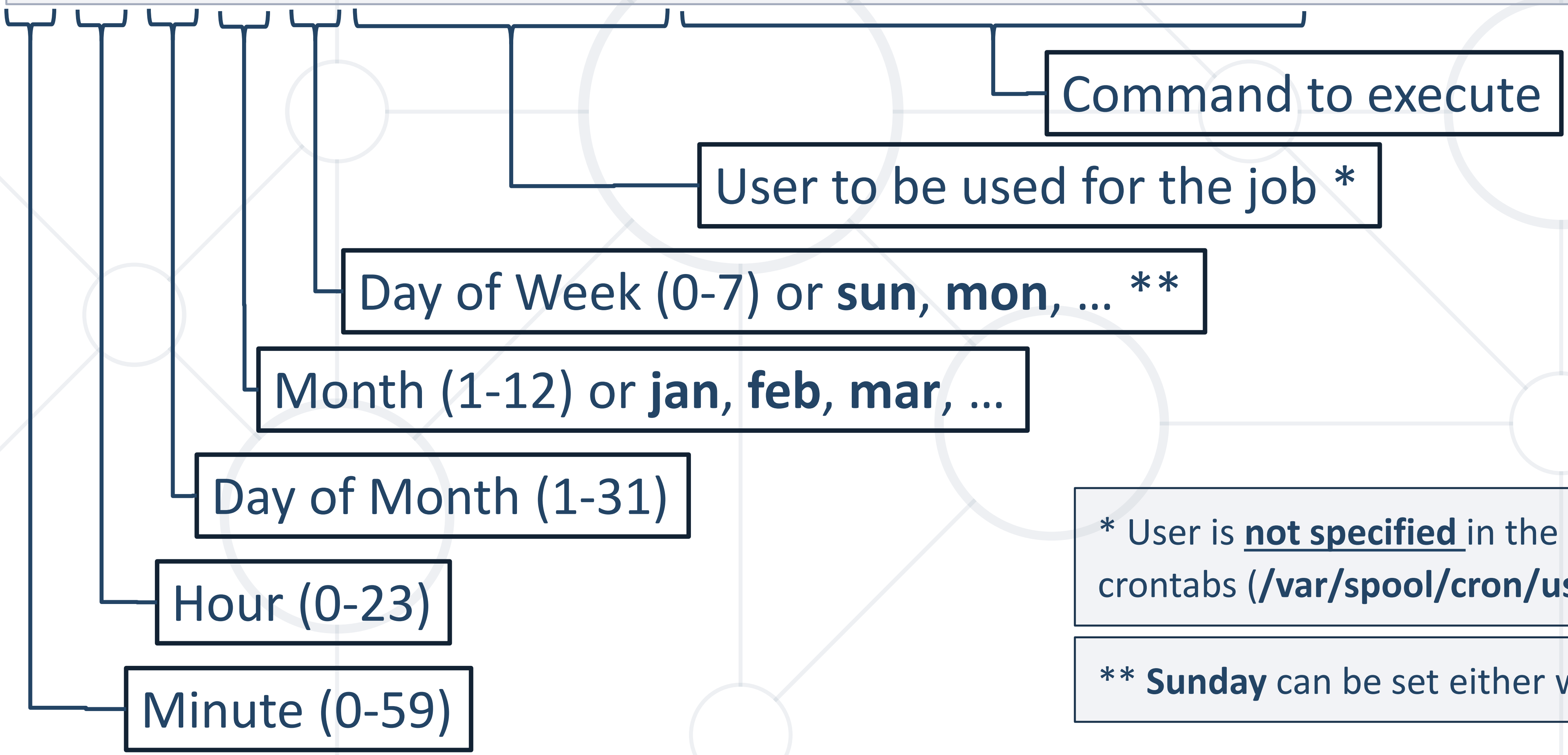
- Regular and repetitive tasks
  - Cleaning, archiving, monitoring, …
- Runtime varies
  - Schedule based or one-time, but at specific moment
- Defined on system or user level

cron Introduction

- **cron** is the main task scheduler in Linux
- Components
  - **crond** => Daemon
  - **crontab** => Management tool
- Configuration files
  - **Tasks** are read from **/etc/crontab** and **/etc/cron.d/***
  - **Rights** are read from **/etc/cron.allow** and **/etc/cron.deny**
- Per user jobs
  - **/var/spool/cron/***

cron Format

```
* * * * * user-name /path/to/executable
```


Command to execute
User to be used for the job *
Day of Week (0-7) or **sun**, **mon**, … **
Month (1-12) or **jan**, **feb**, **mar**, …
Day of Month (1-31)
Hour (0-23)
Minute (0-59)

* User is **not specified** in the per user crontabs (**/var/spool/cron/user**)

** **Sunday** can be set either with **0** or **7**

cron Examples

```
# Execute every minute
* * * * * /utilities/backup.sh
# Run every noon at 12:00
0 12 * * * /utilities/backup.sh
# Run on 1st of January at 00:00
0 0 1 1 * /utilities/backup.sh
# Run every Monday at 5:30
30 5 * * 1 /utilities/backup.sh
# Run at 00:00 and at 12:00 every day
0 0,12 * * * /utilities/backup.sh
# Run every two hours every day
0 */2 * * * /utilities/backup.sh
# Run hourly between 9 and 17 o'clock every day
0 9-17 * * * /utilities/backup.sh
```

cron Shortcuts

- **@yearly or @annually**
  - Run once a year at midnight of 1st of January (**0 0 1 1 ***)
- **@monthly**
  - Run once a month at midnight of the first day (**0 0 1 * ***)
- **@weekly**
  - Run once a week at midnight on Sunday morning (**0 0 * * 0**)
- **@daily or @midnight**
  - Run once a day at midnight (**0 0 * * ***)
- **@hourly**
  - Run once an hour at the beginning of the hour (**0 * * * ***)

anacron

- Runs commands periodically with **frequency in days**
- It does not assume that the machine is non-stop powered
- For each job **anacron** checks if it has been executed in the last **N** days, where **N** is the interval specified for the job
- Jobs are stored at **/etc/anacrontab**
- Configuration can be tested with **anacron -T**
- Shortcuts – **@daily** or **1**, **@weekly** or **7**, and **@monthly** or **30**

at

- Run a task once at a specific time (you may need to install **at** package)
- Each task is queued at **/var/spool/at**
- Security is defined through **/etc/at.allow** and **/etc/at.deny**
- Tools
  - **at** => Main utility
  - **batch** => Auxiliary utility can be used as at to schedule commands
  - **atq** => Show jobs at **at**'s queue
  - **atrm** => Delete at jobs
- Shortcuts – **today**, **midnight**, **noon**, **teatime**, **date**, **now + time unit**

systemd timer

- Systemd unit files (**.timer**) that control services (**.service**)
- Read from the **same paths** as the other units
- Offer built-in support for **calendar** and **monotonic** events
- Calendar (realtime) timers work the same way as cron jobs
- Monotonic timers activate after a time span relative to a point
- Can be created as **transient** (temporary/on the fly) units as well
- Can be used as an **alternative to cron**

systemd (calendar) timer

/etc/systemd/system/free-mem.timer

```
[Unit]
Description=Runs a service every
day at 04:00

[Timer]
OnCalendar=*-*-* 4:00:00
Persistent=true

[Install]
WantedBy=timers.target
```

/etc/systemd/system/free-mem.service

```
[Unit]
Description=Logs system free
memory
Wants=free-mem.timer

[Service]
Type=oneshot
ExecStart=/usr/bin/free

[Install]
WantedBy=multi-user.target
```

1) We can have more than one **OnCalendar** item

2) **Persistent=true** enables immediate execution after activation if it missed the last start time (if the system was off)

https://www.freedesktop.org/software/systemd/man/systemd.timer.html

OnCalendar

- Has the following format
  - **DayOfWeek Year-Month-Day Hour:Minute:Second**
- DayOfWeek can be specified as **Mon**, **Monday**, **mon**, or **monday**
- There are some special expressions, for example:
  - **monthly** -> *-*-01 00:00:00
  - **weekly** -> Mon *-*-* 00:00:00)
- We can test expressions with **systemd-analyze calendar**

https://www.freedesktop.org/software/systemd/man/systemd.time.html

systemd (monotonic) timer

/etc/systemd/system/free-mem.timer

```
[Unit]
Description=Runs weekly and on
boot

[Timer]
OnBootSec=10min
OnUnitActiveSec=1w

[Install]
WantedBy=timers.target
```

/etc/systemd/system/free-mem.service

```
[Unit]
Description=Logs system free
memory
Wants=free-mem.timer

[Service]
Type=oneshot
ExecStart=/usr/bin/free

[Install]
WantedBy=multi-user.target
```

1) Other options are **OnActiveSec**, **OnStartupSec**, and **OnUnitInactiveSec**

2) Expressions can be tested with **systemd-analyze timespan**

https://www.freedesktop.org/software/systemd/man/systemd.timer.html

systemd (transient) timer

- Starting an arbitrary command

```
systemd-run --on-active=30 /bin/touch /tmp/file
```

- Starting an existing service unit

```
systemd-run --on-active="6h 15m" --unit free-mem.service
```

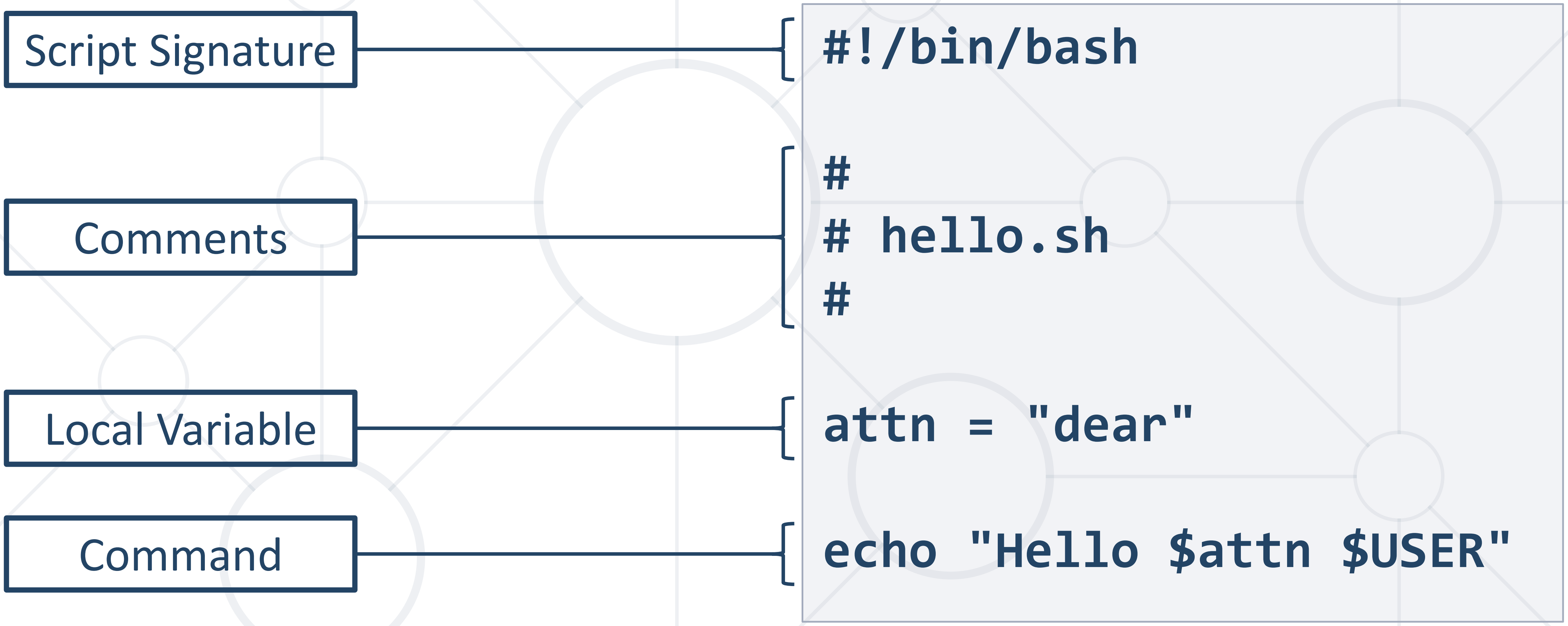
https://wiki.archlinux.org/title/Systemd/Timers

Practice

Scripts

Structure and Variables

Structure



```
#!/bin/bash

#
# hello.sh
#

attn = "dear"

echo "Hello $attn $USER"
```

- **Execution**: `bash hello.sh` or `./hello.sh` or just `hello.sh`

Commands and Flow Control #1

echo

- Description
  - Display line of text
- Example

```
[user@host ~]$ echo 'Hello world!'
Hello world!


[user@host ~]$ echo 'Current user: '$USER
Current user: user
```

printf

- Description
  - Formats and prints text
- Example

```
[user@host ~]$ printf 'Hello world!\n'
Hello world!


[user@host ~]$ printf 'I say %d is the answer\n' 42
I say 42 is the answer
```

seq

- Description
  - Count from starting to ending point
- Example

```
$ seq 1 5
1 2 3 4 5
$ seq 1 2 5
1 3 5
$ seq -w 5 10
05 06 07 08 09 10
```

for (1)

- Description
  - Execute command for each member in a list
- Example

```
# List all files with prefix “item:”
for i in $( ls ); do
   echo item: $i
done
# Create files fileXX.txt where XX is between 05 and 10
for i in $( seq -w 5 10 ); do
   touch file$i.txt
done
```

for (2)

```
# Iterate over the elements of a list
for i in {1..10}; do
   echo item: $i
done
# C-style for loop
for ((i=1;i<=10;i++)); do
   echo item: $i
done
# Nested loops
for i in {1..10}; do
   for j in {1..10}; do
      echo $i-$j
   done
done
```

Practice

Commands and Flow Control #2

test

- Description
  - Evaluate conditional expression
- Example

```
# Compare numbers: OP1 -eq|-ne|-lt|-le|-gt|-ge OP2

# Compare strings: ST1 =|!=|<|> ST2

# Compare files: FL1 -nt|-ot FL2

# File tests: -d|-e|-f|-x FILE
```

if

- Description
  - Execute commands based on conditional
- Example

```
count=1
if [ $count -eq 0 ]; then
  echo 'Equal to 0'
else
  echo 'Not equal to 0'
fi
```

while

- Description
  - Execute commands as long as a test succeeds
- Example

```
# Print numbers from 1 to 5
count=1
while [ $count -le 5 ]; do
  echo $count
  count=$((count+1))
done
```

until

- Description
  - Execute commands as long as a test does not succeed
- Example

```
# Print numbers from 1 to 5
count=1
until [ $count -gt 5 ]; do
  echo $count
  count=$((count+1))
done
```

case

- Description
  - Execute commands based on conditional
- Example

```
count=1
case $count
 1) echo 'One'
    ;;
 *) echo 'Not one'
esac
```

Scripts with Parameters and Prompts

Special Variables

- Name of the script $0
- Positional arguments $1 .. $9, ${10}, ${11} …
- Total number of arguments $#
- List of positional parameters $* or $@
- Exit code of last executed command $?

read

- Description
  - Read a line from the standard input and split it into fields
- Example

```
[user@host ~]$ read -p "Enter name:" NM_ENT
Enter name: James


[user@host ~]$ echo $NM_ENT
James
```

Work with Prompt

- Interactive prompt for user input

```
#!/bin/bash

# Ask for user input

read -p 'Enter your name: ' USR_NAME

echo 'Hello, '$USR_NAME
```

Accept One Parameter

- Check and accept just one parameter

```
#!/bin/bash

# Accept one parameter

if [ $# -ne 1 ]; then
  echo 'Usage: '$0' your_name';
  exit 1;
fi

echo 'Hello, '$1
```

Practice

Summary

- Sourcing is an alternative approach to script execution
- Sourcing can be done in two ways source script.sh or . script.sh
- cron and at are tools for scheduling tasks execution
- Systemd times can be used to schedule tasks as well

Summary

- Bash scripts are built from comments (#) and commands
- Bash scripts can accept parameters on the command line and user input
- We can use flow-control (if, case) and loop (for, while, until) commands

Resources

- Bash Programming – Introduction How-To
  - http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html
- Bash Reference Manual
  - https://www.gnu.org/software/bash/manual/html_node/index.html
- Cron How-To
  - https://help.ubuntu.com/community/CronHowto
- Cron Schedule Expressions Editor
  - https://crontab.guru/

Questions?

Software University

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

СофтУни БУДИТЕЛ

Finance Academy

SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

top EMPLOYER България Bulgaria 2024 FOR A BETTER WORLD OF WORK

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

INDEAVR
Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License