New option inside mshell: working with pdf files.

mshell: (https://www.appservgrid.com/paw92/index.php/2025/03/30/mshell-new-linux-shell-for-ai-and-mathematics/) + local LLMs or Linux LLM distributed evaluation framework), ubuntu 24.04 LTS.

ollamaNexec and ollamaN are internal commands for mshell – Linux shell for AI and mathematics. Here are examples of simple ms scripts to define main characteristics of computers, make declarative diagnostics and troubleshooting of functionality. Throw ollamaNexec commands you also can create and execute single/multiple commands, python code with libraries support, bash scripts to utilize different tasks. You can create ms scripts (internal support inside mshell). Also, you can integrate it with mel-editor that support direct pipes to mshell and mshell commands. Software is working on all linux environments, like Ubuntu or Debian and Raspberry PI OSes)

You can install separate user on the user's system, (container, farm, k8s/ k3s nodes, etc.) with a shell like mshell and it'll be ready to go with couple lines of code. Here are examples new option for ollamaNexec to work with pdf content.

Content:

---

1. Syntax of new option.

- ### Main syntax of commands:

ollamaNexec “some prompt that includes the name and location of file or just name if file is at current directory” – read pdf file, analyze it, run selected single/multiple commands, bash scripts, ms scripts, python code.

Example:

/home/igor > ollama3exec "Use file linuxcommands.pdf at current directory. Open files and create examples for 4 first commands from '1. File and Directory Management' Write bash script that use these 4 first commands."

- ### Mshell’s command for information support:

ollamaN “some prompt that describe task and includes the name and location of pdf file or just name if file is at current directory”

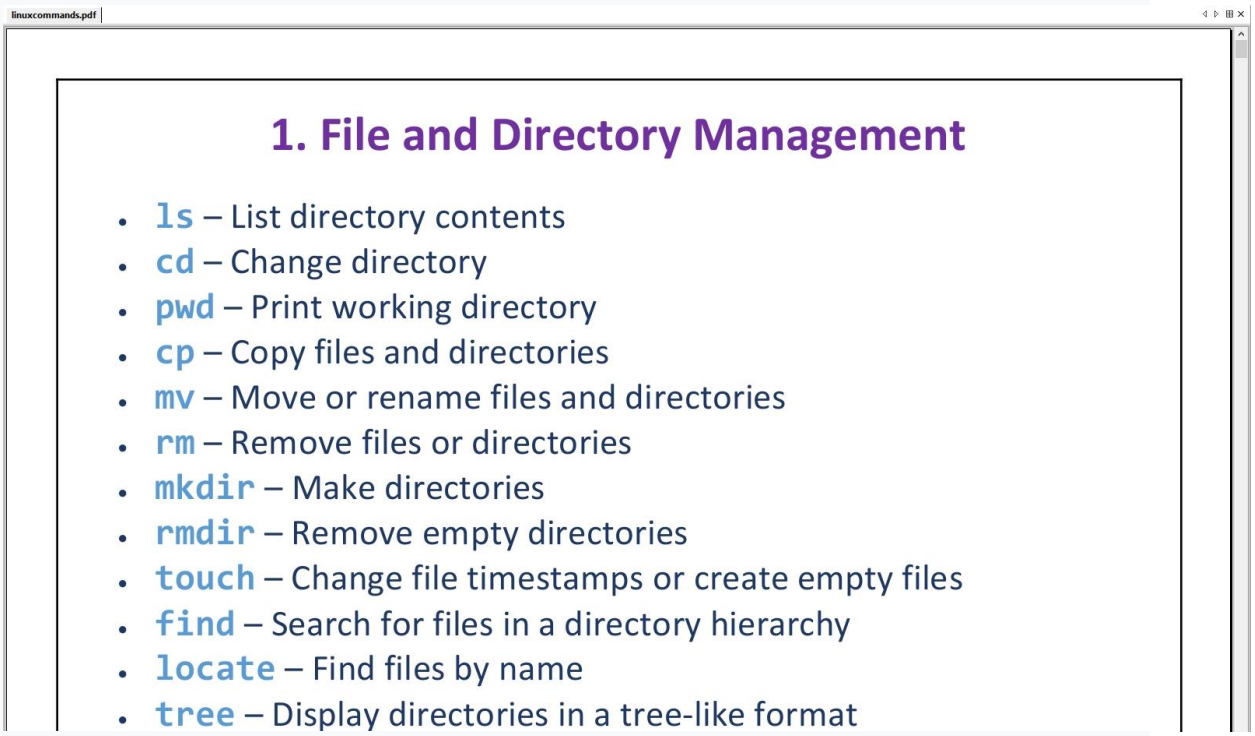
Example:

/home/igor > ollama1 “analyze pdf file /home/igor/some_name.pdf and find what options has df ubuntu linux command and build mermaid diagram with explanation for options”

- P.S.: The code execution during execution occurs after the message inside terminal messages like: **"Executing…"** and highlighted at this note with **blue color.**

2. Example 1: Analyze pdf file and run commands from it.

You have some pdf file with content like single/multiple linux commands, bash or ms scripts, python code, for example like this one:



Throw mshell CLI use internal command ollamaNexec. Mshell will find necessary content inside pdf file and execute it at context that you need (At this

case it’l make next operations: ### Explanation: 1.) **List Directory Contents (`ls -la`) **: This command lists all files and directories in the current directory, including hidden ones.

2.) **Change Directory (`cd /etc`)**: This command changes the current working directory to `/etc`.

3.) **Print Working Directory (`pwd`)**: This command prints the current working directory.

4.) **Copy Files and Directories (`cp /etc/passwd /tmp/passwd_copy.txt`)**: This command copies the `/etc/passwd` file to `/tmp/passwd_copy.txt`.

/home/igor > ollama1exec "Use file linuxcommands.pdf at current directory. Open files and create examples for 4 first commands from '1. File and Directory Management' Write bash script that use these 4 first commands."

Found file: linuxcommands.pdf (0.63 MB)

Processing large document...

Processing potentially large document: linuxcommands.pdf

Detected PDF file - attempting text extraction...

Extracting text from PDF using pdftotext...

Successfully extracted 20.35 KB of text from PDF

Included LARGE DOCUMENT content: linuxcommands.pdf (0.02 MB)

Scanning query for file references (supporting up to 100MB PDF/DOC files)...

=== FILE INCLUSION SUMMARY ===

Total files auto-included: 1

Large documents (PDF/DOC): 1

Enhanced query size: 0.02 MB (21968 bytes)

LLM will now have access to all included document content

==========

=== QUERY TO LLM DEBUG ===

Final query length: 21968

Full query:

[Use file linuxcommands.pdf at current directory. Open files and create examples for 4 first commands from '1. File and Directory Management' Write bash script that use these 4 first commands.

==========

==========

LARGE DOCUMENT AUTO-INCLUDED

==========

==========

Document: linuxcommands.pdf

Content size: 0.02 MB (21235 bytes)

==========

==========

==========

==

PDF TEXT CONTENT EXTRACTED

1. File and Directory Management

ls \u2013 List directory contents

cd \u2013 Change directory

pwd \u2013 Print working directory

cp \u2013 Copy files and directories

mv \u2013 Move or rename files and directories

rm \u2013 Remove files or directories

mkdir \u2013 Make directories

rmdir \u2013 Remove empty directories

touch \u2013 Change file timestamps or create empty files

find \u2013 Search for files in a directory hierarchy

locate \u2013 Find files by name

tree \u2013 Display directories in a tree-like format

chmod \u2013 Change file permissions

chown \u2013 Change file owner and group

chgrp \u2013 Change group ownership

stat \u2013 Display file or file system status

2. File Viewing and Editing

cat \u2013 Concatenate and display file content

tac \u2013 Concatenate and display file content in reverse

more \u2013 View file content interactively (page by page)

less \u2013 View file content interactively (scrollable)

head \u2013 Output the first part of a file

tail \u2013 Output the last part of a file

nano \u2013 Text editor (terminal-based)

vim / vi \u2013 Advanced text editors

emacs \u2013 Text editor

grep \u2013 Search text using patterns

sed \u2013 Stream editor for filtering and transforming text

awk \u2013 Pattern scanning and processing language

cut \u2013 Remove sections from each line of files

sort \u2013 Sort lines of text files

uniq \u2013 Report or omit repeated lines

3. Process Management

ps \u2013 Report a snapshot of current processes

top \u2013 Display Linux tasks

htop \u2013 Interactive process viewer (advanced top)

kill \u2013 Send a signal to a process, typically to terminate

killall \u2013 Terminate processes by name

bg \u2013 Resume a suspended job in the background

fg \u2013 Bring a job to the foreground

jobs \u2013 List active jobs

nice \u2013 Run a program with modified scheduling priority

renice \u2013 Alter priority of running processes

uptime \u2013 Show how long the system has been running

time \u2013 Measure program running time

4. Disk Management

df \u2013 Report file system disk space usage

du \u2013 Estimate file space usage

fdisk \u2013 Partition table manipulator for Linux

lsblk \u2013 List information about block devices

mount \u2013 Mount a file system

umount \u2013 Unmount a file system

parted \u2013 A partition manipulation program

mkfs \u2013 Create a file system

fsck \u2013 File system consistency check and repair

blkid \u2013 Locate/print block device attributes

5. Networking

ifconfig \u2013 Configure network interfaces

ip \u2013 Show/manipulate routing, devices, and tunnels

ping \u2013 Send ICMP Echo requests to network hosts

netstat \u2013 Network statistics

ss \u2013 Socket statistics (faster than netstat)

traceroute \u2013 Trace the route packets take to a network host

nslookup \u2013 Query Internet name servers interactively

dig \u2013 DNS lookup utility

wget \u2013 Non-interactive network downloader

curl \u2013 Transfer data with URLs

scp \u2013 Secure copy files between hosts

ssh \u2013 Secure shell for remote login

ftp \u2013 File Transfer Protocol client

6. User and Group Management

useradd \u2013 Add a user to the system

usermod \u2013 Modify a user account

userdel \u2013 Delete a user account

groupadd \u2013 Add a group to the system

groupdel \u2013 Delete a group

passwd \u2013 Change user password

chage \u2013 Change user password expiry information

whoami \u2013 Print the current logged-in user

who \u2013 Show who is logged in

w \u2013 Show who is logged in and what they\u2019re doing

id \u2013 Display user and group information

groups \u2013 Show user\u2019s groups

7. System Information and Monitoring

uname \u2013 Print system information

hostname \u2013 Show or set the system\u2019s hostname

uptime \u2013 How long the system has been running

dmesg \u2013 Boot and system messages

free \u2013 Display memory usage

top \u2013 Display Linux tasks

vmstat \u2013 Report virtual memory statistics

lscpu \u2013 Display information about the CPU architecture

lsusb \u2013 List USB devices

lspci \u2013 List PCI devices

lshw \u2013 List hardware configuration

8. Archiving and Compression

tar \u2013 Archive files

o tar -czf archive.tar.gz /path/to/directory \u2013 Compress files using gzip

o tar -xzf archive.tar.gz \u2013 Extract gzipped tarball

o tar -cf archive.tar /path/to/directory \u2013 Create a tarball

o tar -xf archive.tar \u2013 Extract tarball

zip \u2013 Package and compress files into a ZIP archive

unzip \u2013 Extract files from a ZIP archive

gzip \u2013 Compress files using the gzip algorithm

gunzip \u2013 Decompress files compressed with gzip

bzip2 \u2013 Compress files using the bzip2 algorithm

bunzip2 \u2013 Decompress files compressed with bzip2

xz \u2013 Compress files using the xz algorithm

unxz \u2013 Decompress files compressed with xz

9. Package Management (Depends on Distribution)

Debian-based (e.g., Ubuntu)

apt-get \u2013 APT package handling utility

o apt-get install <package> \u2013 Install a package

o apt-get update \u2013 Update package list

o apt-get upgrade \u2013 Upgrade installed packages

o apt-get remove <package> \u2013 Remove a package

apt-cache \u2013 Query APT cache

o apt-cache search <package> \u2013 Search for a package

o apt-cache show <package> \u2013 Show package details

Red Hat-based (e.g., CentOS, Fedora)

yum \u2013 Package manager for RPM-based systems

o yum install <package> \u2013 Install a package

o yum update \u2013 Update installed packages

o yum remove <package> \u2013 Remove a package

dnf \u2013 Next-generation package manager (Fedora, CentOS 8+)

o dnf install <package> \u2013 Install a package

o dnf update \u2013 Update installed packages

o dnf remove <package> \u2013 Remove a package

General Commands

rpm \u2013 RPM package manager

o rpm -i <package.rpm> \u2013 Install an RPM package

o rpm -e <package> \u2013 Remove an RPM package

dpkg \u2013 Debian package manager

o dpkg -i <package.deb> \u2013 Install a Debian package

o dpkg -r <package> \u2013 Remove a Debian package

10. System Services and Daemon Management

systemctl \u2013 Control the systemd system and service manager

o systemctl start <service> \u2013 Start a service

o systemctl stop <service> \u2013 Stop a service

o systemctl restart <service> \u2013 Restart a service

o systemctl enable <service> \u2013 Enable a service to start on boot

o systemctl disable <service> \u2013 Disable a service from starting on boot

o systemctl status <service> \u2013 Check service status

service \u2013 Older service management command (used in nonsystemd systems)

o service <service> start \u2013 Start a service

o service <service> stop \u2013 Stop a service

o service <service> restart \u2013 Restart a service

o service <service> status \u2013 Check service status

11. Scheduling Tasks

cron \u2013 Daemon for running scheduled commands

o crontab -e \u2013 Edit cron jobs for the current user

o crontab -l \u2013 List the current user\u2019s cron jobs

o crontab -r \u2013 Remove the current user's cron jobs

at \u2013 Run commands at a specified time

o at 09:00 \u2013 Schedule a command to run at 09:00 AM

batch \u2013 Run commands when the system load is low

sleep \u2013 Delay for a specified time

o sleep 5s \u2013 Sleep for 5 seconds

12. File Permissions and Security

chmod \u2013 Change file permissions

chown \u2013 Change file owner and group

chgrp \u2013 Change the group ownership of a file

umask \u2013 Set default permissions for new files

setfacl \u2013 Set file access control lists (ACL)

getfacl \u2013 Get file access control lists (ACL)

sudo \u2013 Execute a command as another user (usually root)

visudo \u2013 Edit the sudoers file safely

passwd \u2013 Change a user\u2019s password

sudoers \u2013 Manage sudo access for users

gpasswd \u2013 Administer group password

ss \u2013 Display socket statistics (for secure network connections)

13. System Backup and Restore

rsync \u2013 Remote file and directory synchronization

o rsync -avz source/ destination/ \u2013 Synchronize files

o rsync -avz -e ssh source/ user@remote:/destination/ \u2013 Sync over SSH

cpio \u2013 Copy files to and from archives

dd \u2013 Low-level copying and backup of entire filesystems

dd if=/dev/sda of=/path/to/backup.img \u2013 Backup a disk/partition

o dd if=/path/to/backup.img of=/dev/sda \u2013 Restore a disk/partition

o

14. System Diagnostics and Troubleshooting

dmesg \u2013 Print the kernel ring buffer messages (system boot and hardware-related messages)

journalctl \u2013 Query and view logs from systemd\u2019s journal

strace \u2013 Trace system calls and signals

o strace <command> \u2013 Trace a command\u2019s system calls

lsof \u2013 List open files (useful for debugging)

o lsof <file> \u2013 Show processes using a specific file

vmstat \u2013 Report virtual memory statistics

iostat \u2013 Report CPU and I/O statistics

mpstat \u2013 Report CPU usage statistics

pidstat \u2013 Report statistics by process

free \u2013 Display memory usage

uptime \u2013 How long the system has been running

watch \u2013 Execute a program periodically, showing output

o watch -n 1 free \u2013 Watch memory usage every second

lshw \u2013 List hardware configuration

htop \u2013 Interactive process viewer (better than top)

netstat \u2013 Network statistics (deprecated in favor of ss)

ss \u2013 Show socket statistics (more efficient than netstat)

15. Networking & Remote Management

ifconfig \u2013 Configure network interfaces (older command,

replaced by ip)

ip \u2013 A more modern alternative for managing network interfaces

and routing

o ip addr \u2013 Show IP addresses

o ip link \u2013 Show or manipulate network interfaces

ip route \u2013 Show or manipulate routing tables

ss \u2013 Display socket statistics (useful for diagnosing network issues)

nmap \u2013 Network exploration tool (can be used for security auditing)

telnet \u2013 User interface to the TELNET protocol (less common nowadays)

nc (Netcat) \u2013 Network utility for reading and writing from network connections

o nc -l -p 1234 \u2013 Listen on port 1234

o nc <host> <port> \u2013 Connect to a host and port

iptables \u2013 Administration tool for IPv4 packet filtering and NAT (Network Address Translation)

firewalld \u2013 Frontend for managing firewall rules (used in some distros like Fedora and CentOS)

ufw \u2013 Uncomplicated firewall (front-end for iptables)

o ufw enable \u2013 Enable firewall

o ufw allow <port> \u2013 Allow traffic on a specific port

tcpdump \u2013 Command-line packet analyzer

curl \u2013 Transfer data from or to a server using various protocols

(HTTP, FTP, etc.)

wget \u2013 Download files from the web via HTTP, HTTPS, FTP

scp \u2013 Secure copy over SSH (used to copy files between systems)

o scp file.txt user@remote:/path/to/destination/ \u2013 Copy file to remote server

rsync \u2013 Remote file and directory synchronization (often used for backups)

o rsync -avz /local/path/ remote:/remote/path/ \u2013 Sync directories

16. Text Processing Utilities

grep \u2013 Search for patterns within files

o grep 'pattern' file.txt \u2013 Search for a pattern in a file

o grep -r 'pattern' /dir/ \u2013 Recursively search for a pattern

sed \u2013 Stream editor for filtering and transforming text

o sed 's/old/new/g' file.txt \u2013 Replace old with new globally

awk \u2013 A powerful text processing language

o awk '{print $1}' file.txt \u2013 Print the first column of each line in a file

cut \u2013 Remove sections from each line of a file

o cut -d ':' -f 1 /etc/passwd \u2013 Print the first field of each line,

delimited by ":"

sort \u2013 Sort lines of text files

o sort file.txt \u2013 Sort file content in ascending order

uniq \u2013 Report or omit repeated lines in a file

o sort file.txt | uniq \u2013 Sort and remove duplicate lines

tee \u2013 Read from standard input and write to standard output and files

o echo "text" | tee file.txt \u2013 Write to file and show output on screen

tr \u2013 Translate or delete characters

o echo "hello" | tr 'a-z' 'A-Z' \u2013 Convert lowercase to uppercase

paste \u2013 Merge lines of files

o paste file1.txt file2.txt \u2013 Combine lines of file1 and file2 side by side

wc \u2013 Word, line, character, and byte count

o wc -l file.txt \u2013 Count lines in a file

o wc -w file.txt \u2013 Count words in a file

17. System Shutdown and Reboot

shutdown \u2013 Shut down the system

o shutdown -h now \u2013 Immediately shut down

o shutdown -r now \u2013 Reboot the system

o shutdown -h +10 \u2013 Shut down after 10 minutes

reboot \u2013 Reboot the system

halt \u2013 Halt the system immediately (equivalent to turning off power)

poweroff \u2013 Power off the system

init \u2013 Change the runlevel (old-style system manager)

o init 0 \u2013 Shutdown

o init 6 \u2013 Reboot

18. File System Mounting and Management

mount \u2013 Mount a file system

o mount /dev/sda1 /mnt \u2013 Mount partition to a directory

umount \u2013 Unmount a file system

o umount /mnt \u2013 Unmount the file system mounted at /mnt

fstab \u2013 File system table (configuration file for mounting file systems)

o /etc/fstab \u2013 View and configure persistent mount points

blkid \u2013 Display block device attributes

fsck \u2013 Check and repair a file system

o fsck /dev/sda1 \u2013 Check and repair /dev/sda1

19. Filesystem Permissions and Security

chmod \u2013 Change file permissions

o chmod 755 file.txt \u2013 Give read, write, and execute permissions to owner, and read-execute permissions to others

chown \u2013 Change file owner and group

o chown user:group file.txt \u2013 Change owner and group of a file

chgrp \u2013 Change group ownership of a file

o chgrp group file.txt \u2013 Change the group of a file

umask \u2013 Set default permissions for new files

o umask 022 \u2013 Set default permissions for newly created files to 755

setfacl \u2013 Set access control lists (ACL) for file permissions

getfacl \u2013 Get access control lists (ACL) for file permissions

20. Containerization and Orchestration

Docker

docker \u2013 Docker command-line interface (CLI) for managing containers

o docker run <image> \u2013 Run a container from an image

o docker ps \u2013 List running containers

o docker ps -a \u2013 List all containers, including stopped ones

o docker build -t <image_name> . \u2013 Build an image from a Dockerfile

o docker exec -it <container_id> bash \u2013 Start an interactive bash shell inside a running container

o docker stop <container_id> \u2013 Stop a container

o docker rm <container_id> \u2013 Remove a container

o docker logs <container_id> \u2013 View logs of a container

o docker images \u2013 List available images

o docker rmi <image_name> \u2013 Remove an image

o docker network ls \u2013 List Docker networks

o docker-compose \u2013 Manage multi-container Docker applications

\uf0a7 docker-compose up \u2013 Start up a multi-container environment

\uf0a7 docker-compose down \u2013 Stop and remove containers created by docker-compose

\uf0a7 docker-compose logs \u2013 View logs from containers managed by docker-compose

Kubernetes (k8s)

kubectl \u2013 Command-line tool for interacting with Kubernetes clusters

o kubectl get pods \u2013 List pods in the current namespace

o kubectl get nodes \u2013 List nodes in the cluster

o kubectl get services \u2013 List services in the cluster

kubectl apply -f <file>.yaml \u2013 Apply configuration

from a file (e.g., a deployment or pod configuration)

o kubectl create -f <file>.yaml \u2013 Create a resource

from a file

o kubectl delete -f <file>.yaml \u2013 Delete a resource

defined in a file

o kubectl exec -it <pod_name> -- bash \u2013 Execute a

command inside a pod (e.g., open a shell)

o kubectl logs <pod_name> \u2013 View the logs of a pod

o kubectl describe pod <pod_name> \u2013 Get detailed

information about a pod

o kubectl scale deployment <deployment_name> -replicas=<number> \u2013

Scale a deployment to the desired

number of replicas

o kubectl rollout restart deployment

<deployment_name> \u2013 Restart a deployment

o kubectl port-forward pod <pod_name>

<local_port>:<remote_port> \u2013 Forward a port from a

pod to localhost

Helm

helm \u2013 Kubernetes package manager for deploying applications

o helm install <release_name> <chart_name> \u2013

Install a Helm chart

o helm upgrade <release_name> <chart_name> \u2013

Upgrade a Helm release

o helm list \u2013 List all Helm releases

o helm delete <release_name> \u2013 Delete a Helm release

o helm search <chart_name> \u2013 Search for a Helm chart

21. Automation and Configuration Management

Ansible

ansible \u2013 Automation tool for configuration management

ansible all -m ping \u2013 Ping all hosts defined in the

inventory

o ansible-playbook playbook.yml \u2013 Run an Ansible

playbook

o ansible -m command -a 'command' <host> \u2013 Run a

single command on a target host

o ansible-playbook --check playbook.yml \u2013 Dry-run

a playbook to see what would change

o ansible-playbook --limit <host> playbook.yml \u2013

Run a playbook on a specific host or group

o ansible-playbook --extra-vars "key=value" \u2013

Pass extra variables to a playbook

o

Terraform

terraform \u2013 Infrastructure as code tool for provisioning and managing cloud resources

o terraform init \u2013 Initialize a working directory for Terraform configuration

o terraform plan \u2013 Show an execution plan (preview of what changes will be made)

o terraform apply \u2013 Apply the changes described in a Terraform configuration

o terraform destroy \u2013 Destroy infrastructure created by Terraform

o terraform validate \u2013 Validate the configuration files

o terraform show \u2013 Show the current state of the infrastructure

Puppet

puppet \u2013 Configuration management tool

o puppet apply <manifest.pp> \u2013 Apply a Puppet

manifest locally

o puppet agent --test \u2013 Test the Puppet agent (can be

used to run a one-off run)

puppet resource \u2013 Show the current state of resources

(files, services, etc.)

22. CI/CD Tools and Commands

Jenkins

jenkins \u2013 Continuous integration tool

o java -jar jenkins.war \u2013 Start Jenkins from a WAR file

o Access Jenkins through http://localhost:8080 by default

GitLab CI

.gitlab-ci.yml \u2013 Configuration file for GitLab CI/CD pipelines

(typically resides in your repository)

o gitlab-runner register \u2013 Register a new runner with

GitLab

o gitlab-runner run \u2013 Run the GitLab Runner to process

jobs

GitHub Actions

GitHub Actions uses YAML configuration files (typically located in .github/workflows/)

o actions/checkout@v2 \u2013 Checkout the repository code in your CI pipeline

o actions/setup-node@v2 \u2013 Setup Node.js for use in a pipeline

o docker/setup-buildx-action@v1 \u2013 Set up Docker Buildx for building multi-platform images

23. Cloud Services

AWS CLI (Amazon Web Services)

aws \u2013 Command-line tool for managing AWS services

o aws configure \u2013 Configure AWS CLI with your credentials

aws s3 cp file.txt s3://bucket-name/ \u2013 Copy a file to an S3 bucket

o aws ec2 describe-instances \u2013 Describe EC2 instances

o aws ec2 start-instances --instance-ids <id> \u2013

Start an EC2 instance

o aws ec2 stop-instances --instance-ids <id> \u2013

Stop an EC2 instance

o aws s3 sync \u2013 Sync directories with an S3 bucket

o

Azure CLI (Microsoft Azure)

az \u2013 Command-line tool for managing Azure services

o az login \u2013 Log in to your Azure account

o az vm list \u2013 List all virtual machines

o az vm start --name <vm_name> --resource-group

<resource_group> \u2013 Start an Azure VM

o az storage blob upload \u2013 Upload files to an Azure

blob storage

o az group create \u2013 Create a new resource group in Azure

Google Cloud SDK (gcloud)

gcloud \u2013 Command-line tool for Google Cloud Platform

o gcloud auth login \u2013 Log in to Google Cloud

o gcloud compute instances list \u2013 List compute

instances

o gcloud compute instances stop <instance_name>

\u2013 Stop a Google Cloud VM instance

o gcloud app browse \u2013 Open the current Google App

Engine application in a browser

24. Logging and Monitoring

Prometheus

prometheus \u2013 Open-source system monitoring and alerting

toolkit

o prometheus \u2013 Start Prometheus server (usually runs as a

service in the background)

o prometheus --config.file=<config_file> \u2013 Start

Prometheus with a specific config file

Grafana

grafana-cli \u2013 Command-line interface for managing Grafana

plugins

o grafana-cli plugins install <plugin-name> \u2013

Install a plugin in Grafana

ELK Stack (Elasticsearch, Logstash, Kibana)

elasticsearch \u2013 Search engine for logging and data analytics

o curl -XGET

'localhost:9200/_cluster/health?pretty' \u2013 Get

cluster health status

logstash \u2013 Server-side data processing pipeline

o logstash -f <config_file> \u2013 Run Logstash with the

specified configuration file

kibana \u2013 Web interface for visualizing Elasticsearch data

o Kibana is generally accessed through a web browser

(http://localhost:5601)

[END OF PDF TEXT CONTENT]

==========

==========

END OF LARGE DOCUMENT: linuxcommands.pdf

==========

==========

]

==========

Certainly! Below is a Bash script that uses the first four commands from '1. File and Directory Management' section of `linuxcommands.pdf`:

```bash
#!/bin/bash

# Command 1: List directory contents (ls)
echo "Listing directory contents:"
ls -la

# Command 2: Change directory (cd)
echo "Changing to /etc directory:"
cd /etc

# Command 3: Print working directory (pwd)
echo "Current working directory:"
pwd

# Command 4: Copy files and directories (cp)
echo "Copying /etc/passwd to /tmp/passwd_copy.txt:"
cp /etc/passwd /tmp/passwd_copy.txt
```

Explanation:

1. **List Directory Contents (`ls -la`)**: This command lists all files and directories in the current directory, including hidden ones.

2. **Change Directory (`cd /etc`)**: This command changes the current working directory to `/etc`.
3. **Print Working Directory (`pwd`)**: This command prints the current working directory.
4. **Copy Files and Directories (`cp /etc/passwd /tmp/passwd_copy.txt`)**: This command copies the `/etc/passwd` file to `/tmp/passwd_copy.txt`.

Running the Script:
1. Save the script to a file, e.g., `run_linux_commands.sh`.
2. Make the script executable:
   ```sh
   chmod +x run_linux_commands.sh
   ```
3. Run the script:
   ```sh
   ./run_linux_commands.sh
   ```

This Bash script will execute the four commands as described and print their outputs to the terminal.

=== EXEC MODE DEBUG ===
Full LLM response:

[Certainly! Below is a Bash script that uses the first four commands from '1. File and Directory Management' section of `linuxcommands.pdf`:

```bash
#!/bin/bash

# Command 1: List directory contents (ls)
echo "Listing directory contents:"
ls -la

# Command 2: Change directory (cd)
echo "Changing to /etc directory:"
cd /etc

# Command 3: Print working directory (pwd)
echo "Current working directory:"
pwd

# Command 4: Copy files and directories (cp)
echo "Copying /etc/passwd to /tmp/passwd_copy.txt:"
cp /etc/passwd /tmp/passwd_copy.txt
```

Explanation:

1. **List Directory Contents (`ls -la`)**: This command lists all files and directories in the current directory, including hidden ones.
2. **Change Directory (`cd /etc`)**: This command changes the current working directory to `/etc`.
3. **Print Working Directory (`pwd`)**: This command prints the current working directory.
4. **Copy Files and Directories (`cp /etc/passwd /tmp/passwd_copy.txt`)**: This command copies the `/etc/passwd` file to `/tmp/passwd_copy.txt`.

Running the Script:
1. Save the script to a file, e.g., `run_linux_commands.sh`.
2. Make the script executable:
   ```sh
   chmod +x run_linux_commands.sh
   ```
3. Run the script:
   ```sh
   ./run_linux_commands.sh
   ```

This Bash script will execute the four commands as described and print their outputs to the terminal.]
Response length: 1366
==========

Found code block starting with ```

Detected language: [bash]

Code block content:

[#!/bin/bash

Command 1: List directory contents (ls)

echo "Listing directory contents:"

ls -la

Command 2: Change directory (cd)

echo "Changing to /etc directory:"

cd /etc

Command 3: Print working directory (pwd)

echo "Current working directory:"

pwd

Command 4: Copy files and directories (cp)

echo "Copying /etc/passwd to /tmp/passwd_copy.txt:"

cp /etc/passwd /tmp/passwd_copy.txt

]

Code block length: 392

=== DECIDING EXECUTION PATH ===

EXECUTION PATH 4: Script execution

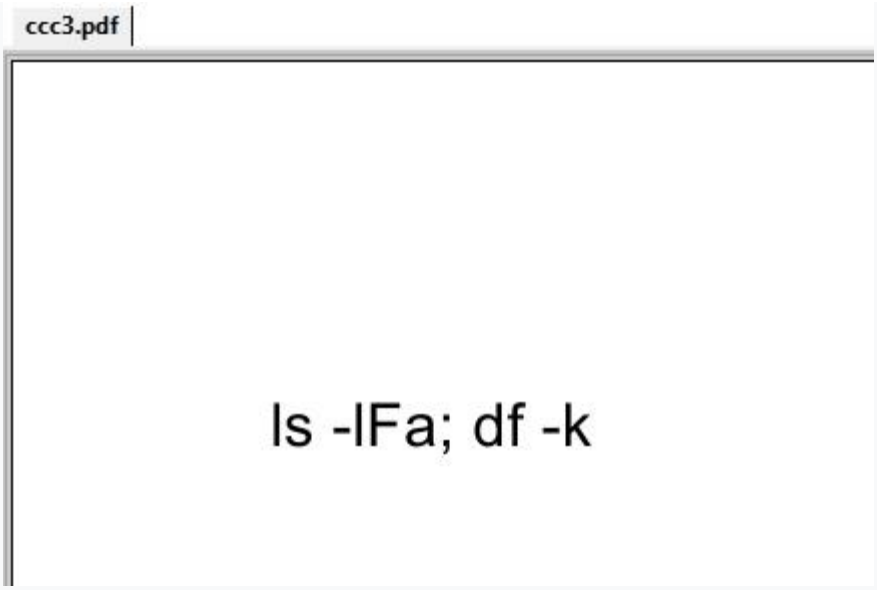
```
Executing as bash script
Listing directory contents:
total 15620
drwxr-x--- 28 igor igor     4096 Sep 25 00:34 .
drwxr-xr-x  6 root root     4096 Feb 23  2025 ..
-rw-------  1 igor igor        5 Mar 15  2025 .bash_history
-rw-r--r--  1 igor igor      220 Feb 23  2025 .bash_logout
-rw-r--r--  1 igor igor     3771 Feb 23  2025 .bashrc
drwxrwxr-x 28 igor igor     4096 Sep 24 03:43 .cache
-rw-rw-r--  1 igor igor     2308 Sep 24 11:03 ccc1.pdf
-rw-rw-r--  1 igor igor      959 Sep 24 11:17 ccc2.pdf
-rw-rw-r--  1 igor igor     1396 Sep 24 15:34 ccc3.pdf
-rw-rw-r--  1 igor igor     2525 Sep 22 15:36 ccc.pdf
drwx------ 20 igor igor     4096 Jun 30 07:11 .config
drwx------  3 igor igor     4096 Mar 15  2025 .dbus
drwxr-xr-x  2 igor igor     4096 Mar 15  2025 Desktop
-rw-r--r--  1 igor igor       25 Mar 15  2025 .dmrc
drwxr-xr-x  2 igor igor     4096 Mar 15  2025 Documents
drwxr-xr-x  3 igor igor     4096 Sep 20 06:05 Downloads
-rwxr--r--  1 igor igor      283 Sep 24 09:34 gc.ms
drwx------  2 igor igor     4096 Jun 20 20:43 .gnupg
drwx------  2 igor igor     4096 Mar 15  2025 .gvfs
-rw-------  1 igor igor        0 Apr 15 22:05 .ICEauthority
drwxrwxr-x  2 igor igor     4096 Mar 15  2025 .ipython
```

```
drwxrwxr-x  3 igor igor     4096 Mar 31 11:52 .jupyter
-rw-------  1 igor igor       20 Sep 23 23:44 .lesshst
-rwxr--r--  1 root root   663554 Sep 25 00:31 linuxcommands.pdf
drwxrwxr-x  6 igor igor     4096 Sep 23 23:47 .local
drwxrwxr-x  5 igor igor     4096 May  2 19:41 melreal
-rwxr--r--  1 root root 14152552 May  2 18:57 mel.zip
drwxrwxr-x  4 igor igor     4096 Sep 22 12:04 mshell
-rw-rw-r--  1 igor igor    30611 Sep 25 00:39 .mshell_filelog
-rwxr--r--  1 igor igor   479153 Sep 25 00:39 .mshellhistory
-rwxr--r--  1 igor igor     1607 Sep 22 15:11 .mshellrc
-rwxr--r--  1 root root   467403 Sep 22 09:38 mshell.zip
drwxr-xr-x  2 igor igor     4096 Mar 15  2025 Music
drwxr-xr-x  2 igor igor     4096 Apr 28 18:40 .ollama
drwxr-xr-x  2 igor igor     4096 Mar 15  2025 Pictures
drwx------  3 igor igor     4096 Mar 31 20:56 .pki
-rw-r--r--  1 igor igor      807 Feb 23  2025 .profile
drwxrwxr-x  7 igor igor     4096 Mar 15  2025 projecto
drwxr-xr-x  2 igor igor     4096 Mar 15  2025 Public
-rw-------  1 igor igor        7 Apr 23 09:04 .python_history
drwxrwxr-x  2 igor igor     4096 May  2 19:01 raznoe
drwxrwxr-x  5 igor igor     4096 Apr 23 11:13 scraping1
drwxrwxr-x  6 igor igor     4096 Apr 17 21:28 scraping1-backups
drwx------  8 igor igor     4096 Jun 20 20:43 snap
drwx------  2 igor igor     4096 Apr  2 00:56 .ssh
```

```
-rw-r--r--  1 igor igor       0 Mar 15  2025 .sudo_as_admin_successful
drwxr-xr-x  2 igor igor    4096 Mar 15  2025 Templates
drwxr-xr-x  2 igor igor    4096 Mar 15  2025 Videos
-rw-------  1 igor igor      50 Sep 23 23:56 .Xauthority
-rw-------  1 igor igor    2894 Sep 23 23:56 .xsession-errors
-rw-------  1 igor igor    2894 Sep 22 13:27 .xsession-errors.old
Changing to /etc directory:
Current working directory:
/etc
Copying /etc/passwd to /tmp/passwd_copy.txt:
=== END EXECUTION PATH ===
=== END EXEC DEBUG ===
/home/igor >
```

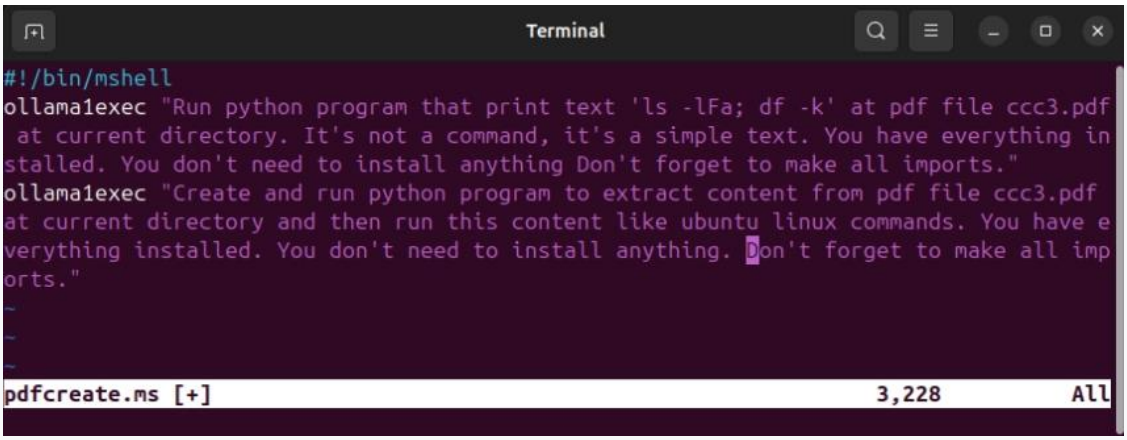
3. Example 2: ms script to create pdf file with content, extract content and run it like script, python code, or single/multiple commands.

You want to create pdf file with some content that you could use at future. This example shows it. It will create pdf file like this with your single/multiple commands, bash or ms scripts, or python code, then extract the code and run it for testing. File will be save at necessary place with code that you want, for example:



Here are an example of simplest ms script to create pdf file, extract info from file and run it, like multiple linux commands:

```
/home/igor > more pdfcreate.ms

#!/bin/mshell

ollama1exec "Run python program that print text 'ls -lFa; df -k' at pdf file ccc

3.pdf at current directory. It's not a command, it's a simple text. You have eve

rything installed. You don't need to install anything. Don't forget to make all

imports."

ollama1exec "Create and run python program to extract content from pdf file ccc3

.pdf at current directory and then run this content like ubuntu linux commands.

You have everything installed. You don't need to install anything. Don't forget

to make all imports."

/home/igor >
```



You can use pipe to mel editor to put output of debug information from ms mshell script to buffer and save it to file later.

Result that was generated by ms mshell script:

/home/igor > ./pdfcreate.ms

Loading .mshellrc...

Loaded environment: PATH=/usr/bin:/bin:/usr/sbin:/usr/local/bin:.:$PATH

Loaded alias: ll='ls -l'

Loaded alias: lsac='ls -la --color=auto'

Loaded alias: home='cd ~'

Loaded environment: EDITOR=nano

Loaded environment: BROWSER=/usr/bin/google-chrome

Loaded environment: PLOTLY_RENDERER=browser

Lua state initialized successfully

Range function defined successfully

Loading history...

=== QUERY TO LLM DEBUG ===

Final query length: 235

Full query:

[Run python program that print text 'ls -lFa; df -k' at pdf file ccc3.pdf at current directory. It's not a command, it's a simple text. You have everything installed. You don't need to install anything! Don't forget to make all imports.]

==========

To create a Python script that writes the text "ls -lFa; df -k" into a PDF file named `ccc3.pdf` in the current directory, you can use the `reportlab` library for creating PDFs. Below is the complete code:

```python
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

# Create a PDF file and write some text to it
def create_pdf(filename, text):
    # Define the size of the page
    c = canvas.Canvas(filename, pagesize=letter)

    # Get the dimensions of the page
    width, height = letter

    # Set font size and position
    font_size = 12
    x, y = 50, height - 75
```

```
    # Write the text to the PDF
    c.setFont("Helvetica", font_size)
    lines = text.split('\n')
    for line in lines:
        c.drawString(x, y, line)
        y -= font_size + 10

    # Save the PDF file
    c.save()

# The text you want to write to the PDF
text_to_write = "ls -lFa; df -k"

# Create the PDF file
create_pdf("ccc3.pdf", text_to_write)
```

Make sure to save this code in a file, for example `create_pdf.py`, and then run it using Python:

```sh
python create_pdf.py
```

This will generate a PDF file named `ccc3.pdf` with the specified text.

=== EXEC MODE DEBUG ===

Full LLM response:

[To create a Python script that writes the text "ls -lFa; df -k" into a PDF file named `ccc3.pdf` in the current directory, you can use the `reportlab` library for creating PDFs. Below is the complete code:

```python
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

# Create a PDF file and write some text to it
def create_pdf(filename, text):
    # Define the size of the page
    c = canvas.Canvas(filename, pagesize=letter)

    # Get the dimensions of the page
    width, height = letter

    # Set font size and position
    font_size = 12
    x, y = 50, height - 75
```

```
    # Write the text to the PDF
    c.setFont("Helvetica", font_size)
    lines = text.split('\n')
    for line in lines:
        c.drawString(x, y, line)
        y -= font_size + 10

    # Save the PDF file
    c.save()

# The text you want to write to the PDF
text_to_write = "ls -lFa; df -k"

# Create the PDF file
create_pdf("ccc3.pdf", text_to_write)
```

Make sure to save this code in a file, for example `create_pdf.py`, and then run it using Python:

```sh
python create_pdf.py
```

This will generate a PDF file named `ccc3.pdf` with the specified text.]

Response length: 1183

==========

Found code block starting with ```

Detected language: [python]

Code block content:

[from reportlab.lib.pagesizes import letter

from reportlab.pdfgen import canvas

```
# Create a PDF file and write some text to it
def create_pdf(filename, text):
    # Define the size of the page
    c = canvas.Canvas(filename, pagesize=letter)

    # Get the dimensions of the page
    width, height = letter

    # Set font size and position
    font_size = 12
    x, y = 50, height - 75

    # Write the text to the PDF
    c.setFont("Helvetica", font_size)
    lines = text.split('\n')
```

```
    for line in lines:
        c.drawString(x, y, line)
        y -= font_size + 10

    # Save the PDF file
    c.save()

# The text you want to write to the PDF
text_to_write = "ls -lFa; df -k"

# Create the PDF file
create_pdf("ccc3.pdf", text_to_write)
]
Code block length: 759
=== DECIDING EXECUTION PATH ===
EXECUTION PATH 1: Python code
Executing Python code block:
=== END EXECUTION PATH ===
=== END EXEC DEBUG ===
Found file: ccc3.pdf (0.00 MB)
Processing large document...
Processing potentially large document: ccc3.pdf
Detected PDF file - attempting text extraction...
Extracting text from PDF using pdftotext...
```

Successfully extracted 0.02 KB of text from PDF

Included LARGE DOCUMENT content: ccc3.pdf (0.00 MB)

Scanning query for file references (supporting up to 100MB PDF/DOC files)...

=== FILE INCLUSION SUMMARY ===

Total files auto-included: 1

Large documents (PDF/DOC): 1

Enhanced query size: 0.00 MB (1301 bytes)

LLM will now have access to all included document content

==========

=== QUERY TO LLM DEBUG ===

Final query length: 1301

Full query:

[Create and run python program to extract content from pdf file ccc3.pdf at current directory and then run this content like ubuntu linux commands. You have everything installed. You don't need to install anything. Don't forget to make all imports.

==========

==========

LARGE DOCUMENT AUTO-INCLUDED

==========

==========

Document: ccc3.pdf

Content size: 0.00 MB (403 bytes)

==========

==========

==========

==

PDF TEXT CONTENT EXTRACTED

==========

==

Original PDF: ccc3.pdf

Extracted text size: 0.02 KB (17 bytes)

Extraction method: pdftotext (poppler-utils)

==========

==

[START OF PDF TEXT CONTENT]

ls -lFa; df -k

[END OF PDF TEXT CONTENT]

==========

==========

END OF LARGE DOCUMENT: ccc3.pdf

==========

==========

]

==========

To extract text from a PDF file and run it as Ubuntu Linux commands, you can use the `pdftotext` utility from `poppler-utils` to extract the text and then execute it using Python's `subprocess` module. Below is the complete code:

```python
import subprocess

# Function to extract text from a PDF file
def extract_text_from_pdf(pdf_file):
    # Use pdftotext command to extract text
    result = subprocess.run(['pdftotext', pdf_file, '-'], capture_output=True, text=True)
    if result.returncode != 0:
        raise Exception(f"Failed to extract text from PDF: {result.stderr}")

    return result.stdout
```

```
# Function to run a command in the shell
def run_command(command):
    # Execute the command using subprocess
    result = subprocess.run(command, capture_output=True, text=True,
shell=True)
    if result.returncode != 0:
        raise Exception(f"Command execution failed: {result.stderr}")

    return result.stdout

# Extract text from the PDF file
pdf_file = "ccc3.pdf"
extracted_text = extract_text_from_pdf(pdf_file)

# Split the extracted text into individual commands
commands = extracted_text.strip().split('\n')

# Run each command and capture the output
outputs = []
for command in commands:
    try:
        output = run_command(command)
        outputs.append(output)
    except Exception as e:
```

```
        outputs.append(f"Failed to execute '{command}': {str(e)}")

# Print all outputs
for output in outputs:
    print(output)
```

Steps to Run the Script

1. Save the script in a file, for example `extract_and_run.py`.
2. Ensure that you have the necessary packages installed:

```sh
pip install pdftotext
sudo apt-get install poppler-utils  # This command installs poppler-utils which includes pdftotext
```

3. Run the script using Python:

```sh
python extract_and_run.py
```

Notes

- The `pdftotext` utility is used to extract text from the PDF file.
- Each line extracted from the PDF is treated as a command and executed using `subprocess.run`.
- Any errors during command execution are caught and printed.

This script assumes that the extracted commands are valid shell commands that can be run on an Ubuntu system. If you encounter any issues, ensure that the extracted text is correctly formatted and that the commands are executable in your environment.

=== EXEC MODE DEBUG ===
Full LLM response:
[To extract text from a PDF file and run it as Ubuntu Linux commands, you can use the `pdftotext` utility from `poppler-utils` to extract the text and then execute it using Python's `subprocess` module. Below is the complete code:

```python
import subprocess

# Function to extract text from a PDF file
def extract_text_from_pdf(pdf_file):
    # Use pdftotext command to extract text
```

```
    result = subprocess.run(['pdftotext', pdf_file, '-'], capture_output=True,
text=True)
    if result.returncode != 0:
        raise Exception(f"Failed to extract text from PDF: {result.stderr}")

    return result.stdout

# Function to run a command in the shell
def run_command(command):
    # Execute the command using subprocess
    result = subprocess.run(command, capture_output=True, text=True,
shell=True)
    if result.returncode != 0:
        raise Exception(f"Command execution failed: {result.stderr}")

    return result.stdout

# Extract text from the PDF file
pdf_file = "ccc3.pdf"
extracted_text = extract_text_from_pdf(pdf_file)

# Split the extracted text into individual commands
commands = extracted_text.strip().split('\n')
```

```
# Run each command and capture the output
outputs = []
for command in commands:
    try:
        output = run_command(command)
        outputs.append(output)
    except Exception as e:
        outputs.append(f"Failed to execute '{command}': {str(e)}")

# Print all outputs
for output in outputs:
    print(output)
```

Steps to Run the Script

1. Save the script in a file, for example `extract_and_run.py`.
2. Ensure that you have the necessary packages installed:

```sh
pip install pdftotext
sudo apt-get install poppler-utils  # This command installs poppler-utils which includes pdftotext
```

3. Run the script using Python:

```sh
python extract_and_run.py
```

Notes

- The `pdftotext` utility is used to extract text from the PDF file.
- Each line extracted from the PDF is treated as a command and executed using `subprocess.run`.
- Any errors during command execution are caught and printed.

This script assumes that the extracted commands are valid shell commands that can be run on an Ubuntu system. If you encounter any issues, ensure that the extracted text is correctly formatted and that the commands are executable in your environment.]
Response length: 2302
==========
Found code block starting with ```
Detected language: [python]
Code block content:
[import subprocess

```
# Function to extract text from a PDF file
def extract_text_from_pdf(pdf_file):
    # Use pdftotext command to extract text
    result = subprocess.run(['pdftotext', pdf_file, '-'], capture_output=True,
text=True)
    if result.returncode != 0:
        raise Exception(f"Failed to extract text from PDF: {result.stderr}")

    return result.stdout

# Function to run a command in the shell
def run_command(command):
    # Execute the command using subprocess
    result = subprocess.run(command, capture_output=True, text=True,
shell=True)
    if result.returncode != 0:
        raise Exception(f"Command execution failed: {result.stderr}")

    return result.stdout

# Extract text from the PDF file
pdf_file = "ccc3.pdf"
extracted_text = extract_text_from_pdf(pdf_file)
```

```
# Split the extracted text into individual commands
commands = extracted_text.strip().split('\n')

# Run each command and capture the output
outputs = []
for command in commands:
    try:
        output = run_command(command)
        outputs.append(output)
    except Exception as e:
        outputs.append(f"Failed to execute '{command}': {str(e)}")

# Print all outputs
for output in outputs:
    print(output)
]
Code block length: 1213
=== DECIDING EXECUTION PATH ===
EXECUTION PATH 1: Python code
Executing Python code block:
total 15648
drwxr-x--- 28 igor igor     4096 Sep 25 12:01 ./
drwxr-xr-x  6 root root     4096 Feb 23  2025 ../
```

```
-rw-------  1 igor igor        5 Mar 15  2025 .bash_history
-rw-r--r--  1 igor igor      220 Feb 23  2025 .bash_logout
-rw-r--r--  1 igor igor     3771 Feb 23  2025 .bashrc
drwxrwxr-x 28 igor igor     4096 Sep 24 03:43 .cache/
-rw-rw-r--  1 igor igor     2308 Sep 24 11:03 ccc1.pdf
-rw-rw-r--  1 igor igor      959 Sep 24 11:17 ccc2.pdf
-rw-rw-r--  1 igor igor     1409 Sep 25 12:01 ccc3.pdf
-rw-rw-r--  1 igor igor      966 Sep 25 10:58 ccc4.pdf
-rw-rw-r--  1 igor igor     2525 Sep 22 15:36 ccc.pdf
drwx------ 20 igor igor     4096 Jun 30 07:11 .config/
drwx------  3 igor igor     4096 Mar 15  2025 .dbus/
drwxr-xr-x  2 igor igor     4096 Mar 15  2025 Desktop/
-rw-r--r--  1 igor igor       25 Mar 15  2025 .dmrc
drwxr-xr-x  2 igor igor     4096 Mar 15  2025 Documents/
drwxr-xr-x  3 igor igor     4096 Sep 20 06:05 Downloads/
-rwxr--r--  1 igor igor      283 Sep 24 09:34 gc.ms*
drwx------  2 igor igor     4096 Jun 20 20:43 .gnupg/
drwx------  2 igor igor     4096 Mar 15  2025 .gvfs/
-rw-------  1 igor igor        0 Apr 15 22:05 .ICEauthority
drwxrwxr-x  2 igor igor     4096 Mar 15  2025 .ipython/
drwxrwxr-x  3 igor igor     4096 Mar 31 11:52 .jupyter/
-rw-------  1 igor igor       20 Sep 23 23:44 .lesshst
-rwxr--r--  1 root root   663554 Sep 25 00:31 linuxcommands.pdf*
drwxrwxr-x  6 igor igor     4096 Sep 23 23:47 .local/
```

```
drwxrwxr-x  5 igor igor     4096 May  2 19:41 melreal/
-rwxr--r--  1 root root 14152552 May  2 18:57 mel.zip*
drwxrwxr-x  4 igor igor     4096 Sep 22 12:04 mshell/
-rw-rw-r--  1 igor igor    37679 Sep 25 12:01 .mshell_filelog
-rwxr--r--  1 igor igor   488411 Sep 25 12:00 .mshellhistory*
-rwxr--r--  1 igor igor     1607 Sep 22 15:11 .mshellrc*
-rwxr--r--  1 root root   467403 Sep 22 09:38 mshell.zip*
drwxr-xr-x  2 igor igor     4096 Mar 15  2025 Music/
drwxr-xr-x  2 igor igor     4096 Apr 28 18:40 .ollama/
-rwxr--r--  1 igor igor      655 Sep 25 11:56 pdfcreate.ms*
drwxr-xr-x  2 igor igor     4096 Mar 15  2025 Pictures/
drwx------  3 igor igor     4096 Mar 31 20:56 .pki/
-rw-r--r--  1 igor igor      807 Feb 23  2025 .profile
drwxrwxr-x  7 igor igor     4096 Mar 15  2025 projecto/
drwxr-xr-x  2 igor igor     4096 Mar 15  2025 Public/
-rw-------  1 igor igor        7 Apr 23 09:04 .python_history
drwxrwxr-x  2 igor igor     4096 May  2 19:01 raznoe/
drwxrwxr-x  5 igor igor     4096 Apr 23 11:13 scraping1/
drwxrwxr-x  6 igor igor     4096 Apr 17 21:28 scraping1-backups/
drwx------  8 igor igor     4096 Jun 20 20:43 snap/
drwx------  2 igor igor     4096 Apr  2 00:56 .ssh/
-rw-r--r--  1 igor igor        0 Mar 15  2025 .sudo_as_admin_successful
drwxr-xr-x  2 igor igor     4096 Mar 15  2025 Templates/
drwxr-xr-x  2 igor igor     4096 Mar 15  2025 Videos/
```

```
-rw-------  1 igor igor       50 Sep 23 23:56 .Xauthority
-rw-------  1 igor igor     2894 Sep 23 23:56 .xsession-errors
-rw-------  1 igor igor     2894 Sep 22 13:27 .xsession-errors.old
Filesystem      1K-blocks      Used  Available Use% Mounted on
tmpfs             2018772     83120    1935652   5% /run
/dev/nvme0n1p5  277647428 219632516   43838400  84% /
tmpfs            10093844    219028    9874816   3% /dev/shm
tmpfs                5120         8       5112   1% /run/lock
efivarfs              192        65        123  35% /sys/firmware/efi/efivars
/dev/nvme0n1p1     262144     34328     227816  14% /boot/efi
tmpfs             2018768       204    2018564   1% /run/user/1001
/dev/sda1      1953453056  45178368 1908274688   3% /media/igor/Backup
Plus
/dev/sdb2      4883466240 934334976 3949131264  20% /media/igor/One
Touch


=== END EXECUTION PATH ===
=== END EXEC DEBUG ===
/home/igor >
```

4. Conclusion.

- Basic description of mshell functionality (Linux compatible OSes like Ubuntu, Debian, etc., Raspberry PI OS) can be found here:

https://www.appservgrid.com/paw92/index.php/2025/03/30/mshell-new-linux-shell-for-ai-and-mathematics/

- The basic principles of how the Linux distributed LLM evaluation framework works can be found in the diagram here: https://www.appservgrid.com/paw92/index.php/2025/03/31/llm-evaluation-framework-for-local-use-may-aug-2024/
- The mshell script + LLM standalone on-premise technology (or Linux LLM distributed evaluation framework) is quite suitable to work with pdf content.

5. Adding information:

- How to make declarative diagnostics, monitoring, troubleshooting and analyze computers with using mshell script and local on-premise LLMs (or Linux LLM evaluation framework): https://www.appservgrid.com/paw92/index.php/2025/05/11/how-to-make-declarative-diagnostics-monitoring-troubleshooting-and-analyze-computers-with-using-mshell-script-and-local-on-premise-llms-or-linux-llm-evaluation-framework/
- Using mshell and local LLMs to create non-scientific publications: https://www.appservgrid.com/paw92/index.php/2025/05/01/using-mshell-and-local-llms-to-create-non-scientific-publications/
- Urban Balcony Garden Design Project example: https://www.appservgrid.com/paw92/index.php/2025/04/29/urban-balcony-garden-design-project-example/
- Nuances of code visualization with different LLMs: https://www.appservgrid.com/paw92/index.php/2025/04/24/nuances-of-code-visualization-with-different-llms/
- Webscraping with mshell and LLM: https://www.appservgrid.com/paw92/index.php/2025/04/22/webscraping-with-mshell-and-llm/
- Visualizations with mshell and LLM's. Local use: https://www.appservgrid.com/paw92/index.php/2025/04/01/visualizations-with-mshell-and-llms-local-use/
- Mel Editor – mini editor for Linux: https://www.appservgrid.com/paw92/index.php/2025/03/30/mel-editor-mini-editor-for-linux/
- 7d Pi-Storage: https://www.appservgrid.com/paw92/index.php/2025/08/20/7d-pi-storage/

Contacts: igor.lukyanov@appservgrid.com Art2Dec Soft Lab.

09.25.2025