ACTA TECHNICA CORVINIENSIS – Bulletin of Engineering | e–ISSN: 2067 – 3809
Tome XIX [2026] | Fascicule 2 [April – June]

[1.] Attila Gábor FERENCZ

OPTIMISED LINUX ISO DEVELOPMENT FOR SMALL AND MEDIUM—SIZED ENTERPRISES

[1.] John von Neumann University, Faculty of Engineering and Computer Science, Kecskemét, HUNGARY

Abstract: In today's IT landscape, Linux—based operating systems are assuming an increasingly prominent role in the design and deployment of enterprise infrastructures, particularly within small and medium—sized enterprises (SMEs). However, the adoption and configuration of Linux systems require a considerable level of technical expertise, which often presents a significant challenge for SMEs, especially under conditions of limited IT resources. Conventional, general—purpose Linux distributions frequently fail to fully meet enterprise—specific requirements, as they do not inherently include the services, configurations, and optimisations necessary for a given organisational context. Consequently, substantial post—installation effort—in terms of both time and resources—is required to customise and optimise the system environment. This issue is particularly pronounced in SMEs, where the rapid and cost—effective deployment of IT infrastructure constitutes a critical factor of competitiveness. A viable solution to these challenges lies in the utilisation of customised Linux ISO images, which enable the creation of pre—configured, standardised system environments tailored to organisational needs. The primary objective of this study is to develop and present a methodological framework for constructing such customised Linux installation packages, specifically designed for SME environments. The research focuses on a concrete implementation scenario, in which an existing Linux distribution—specifically Kubuntu 24.04—is modified to produce a system aligned with enterprise requirements. This work contributes to expanding the applicability of Linux—based solutions in corporate contexts and provides practical, implementation—oriented approaches that support more efficient, secure, and cost—effective IT operations, particularly for small and medium—sized enterprises.
Keywords: Linux operating systems, Kubuntu 24.04, System optimisation for SMEs

INTRODUCTION

The Role of Linux in the Effective Operation of Enterprises

In recent years, there has been a marked increase in interest in the application of the Linux operating system within real—time environments, particularly in control systems. The simplicity and architectural elegance of Linux ensure robustness and high performance, while its open—source licensing model enables extensive modification of the source code to meet specific user requirements (Scordino and Lipari, 2006).

Linux is widely recognised as a stable and reliable foundation for operating system development. Moreover, the creation of a custom Linux distribution represents a highly rewarding technical endeavour, facilitating a deeper understanding of operating system design principles (Koutoupis, 2018). Despite these advantages, Linux was originally designed as a general—purpose operating system, which introduces certain limitations in real—time contexts. These include unpredictable latency, limited support for deterministic scheduling, and coarse—grained timing resolution—factors that can hinder its applicability in real—time systems. Consequently, numerous kernel—level enhancements have been proposed to extend Linux with real—time capabilities (Santana, 2025).

In the contemporary landscape of personalised computing, a standard Linux installation may not adequately satisfy diverse user requirements. The development of a custom Linux ISO can significantly enhance both usability and deployment efficiency. A Linux ISO is a disk image containing all components required for system installation, including the kernel, software packages, and filesystem structure. Through ISO customisation, it becomes possible to create installation media that incorporates predefined applications, configurations, and system settings, thereby streamlining the deployment process (Abbott, 2018).

Custom Linux ISO images enable the pre—integration of workflow—specific applications and configurations, ensuring consistency across multiple system deployments. This approach reduces installation time by embedding all necessary components directly into the installation medium. Furthermore, it facilitates the creation of a tailored computing environment, ultimately resulting in a more efficient and user—centric Linux experience (Yadav, 2024).

At the same time, the capabilities of embedded system hardware continue to expand. However, mainstream Linux distributions often include a wide range of applications and services with substantial resource demands, which can limit their applicability on constrained hardware platforms. The development of optimised Linux ISO images for general users involves tailoring distributions to enhance performance, usability, and hardware compatibility while minimising unnecessary resource consumption.

This process typically focuses on the selection of lightweight desktop environments, pre–configuration of device drivers, and optimisation of core system libraries and the kernel—often leveraging stable, well–established base distributions as a foundation (Díaz et al., 2019).

Linux Installation Requires Advanced Technical Expertise

Linux installation is the process of deploying the Linux operating system onto a computer system, encompassing the validation and modification of installation parameters, copying of system files, persistence of configuration settings, installation of the boot manager, and preparation of the system for initial startup (https://www.sciencedirect.com /topics/computer–science/linux–installation).

A Linux distribution integrates the Linux kernel with a comprehensive set of software packages, thereby providing a fully functional operating system along with most commonly required applications upon installation (Sinclair, 2011). The selection of an appropriate distribution necessitates consideration of multiple factors, including system performance, community support, repository availability, ease of configuration, and operational reliability.

Robust repository support significantly simplifies system management, whereas limited support may require manual package sourcing or compilation from source code (Polstra, 2015). Distributions such as Ubuntu, characterised by a regular release cycle and long–term support (LTS) options, represent a compelling choice for a broad user base (Gediya et al., 2019).

Compared to operating systems such as Windows or macOS, Linux installation and configuration demand a higher level of technical proficiency. The primary advantages of Linux are most effectively realised by technically skilled users who leverage access to the system's source code for customisation and optimisation (Dempster, 2001). Successful deployment and operation often require substantial expertise, and several factors—such as inconsistent technical documentation, limited conventional vendor support, and reduced vendor accountability—have historically impeded widespread adoption within the business sector.

Nevertheless, the full availability of the Linux kernel source code, combined with a forward–looking licensing model, has served as a key driver of Linux's sustained success over the past decades. This ecosystem has fostered a global community of experts who continuously develop and integrate valuable functionalities into the platform (Ganney et al., 2020). Linux remains universally accessible—available to individuals as well as commercial and non–commercial organisations alike—and all versions, including both long–term stable releases and non–LTS variants, can be used without licensing costs.

Advantages of ISO Virtual Disk Image Files in Small and Medium–Sized Enterprises

An ISO image is a disk image of an optical medium, representing an archive file that contains all data intended for an optical disc, stored sector–by–sector, including the associated filesystem. ISO images can be generated from physical optical media using disk imaging software, constructed from collections of files via disc authoring tools, or derived through conversion from other image formats. Software distributed on bootable media is frequently made available in ISO format. Notably, ISO images are uncompressed and do not rely on proprietary container structures; instead, they constitute binary representations of optical media data, preserving the exact sector–level layout. Typically, they encapsulate filesystems such as ISO 9660 (and its extensions) or UDF, including all file data in its original binary form as stored on the medium (Whittaker, 2025).

Most software tools capable of writing ISO images to hard drives or writable optical media (CD/DVD/Blu–ray) are not inherently designed to deploy such images directly onto USB flash drives. This limitation is primarily attributable to tooling constraints rather than deficiencies of the ISO format itself. However, since approximately 2011, a variety of software solutions have emerged that support writing raw disk images to USB flash devices (Yadav, 2024).

ISO–based virtual disk image files (commonly in formats such as *.iso*, *.vmdk*, or *.vhd*) offer substantial advantages for small and medium–sized enterprises by reducing hardware dependency, lowering IT maintenance costs, and enhancing operational flexibility. Through virtualisation technologies, SMEs can establish virtualised IT infrastructures—including email services, file sharing systems, and security platforms—on relatively modest hardware, thereby enabling competitiveness with larger organisations (Aljabari, 2012).

Empirical observations by Cashwell (2024) highlight several key benefits of virtual disk images in SME environments. Cost efficiency is significantly improved, as virtualisation enables multiple virtual machines to operate on a single physical server, thereby reducing expenditures related to hardware acquisition, energy consumption, and cooling. Furthermore, virtual disk images facilitate rapid backup and recovery of complete IT environments compared to traditional physical server–based approaches. They are widely utilised for software deployment, live system environments, and testing scenarios, accelerating IT provisioning cycles. Virtual Desktop Infrastructure (VDI) solutions based on disk images allow employees to access systems and data remotely, supporting flexible and distributed work models.

At the same time, while features such as snapshotting and consistent system state preservation enhance reliability, the centralisation of enterprise data within virtual disk images introduces potential security risks, particularly concerning unauthorised access or data exfiltration. Consequently, robust security mechanisms are essential. Virtualisation has rapidly become a preferred paradigm for organisations seeking to minimise energy and hardware costs while maximising resource utilisation and sharing. Today, system virtualisation is widely adopted across both server and desktop environments, enabling multiple virtual machines to run concurrently on a single physical platform (Alawadh, 2021).

As noted by Sahoo et al. (2010), system virtualisation allows multiple virtual machines—each running distinct operating systems and applications—to execute simultaneously on the same physical hardware. In such environments, operating systems and applications remain abstracted from the underlying hardware, perceiving computational resources as shared, virtualised resource pools. Within this paradigm, the term *guest* refers to the virtual machine, while *host* denotes the underlying physical or virtualised infrastructure providing the execution environment.

METHODS

This study presents a procedure for the development of a deployment—ready installation package tailored to small and medium—sized enterprise environments. The system is based on Kubuntu 24.04, whose software package composition was extended with applications and services specifically relevant to organisational requirements.

Installation of Required Tools for SME—Oriented ISO Image Creation

In order to generate a bootable installation image from the modified file system at the final stage of the customisation process, the installation of two essential software components is required. The first is the ISOLINUX bootloader, which enables the booting of floppy disk images, CD boot sectors, and Linux kernels. This functionality ensures that the customised disk image becomes fully bootable.

As part of the SYSLINUX project, ISOLINUX was specifically designed to facilitate the creation of bootable ISO images from directory structures (Stevens, 2026). The second required tool is *xorriso*, which is capable of copying file objects from POSIX—compliant file systems into Rock Ridge—enhanced ISO 9660 file systems. This capability allows for the accurate generation and structuring of the final installation image (Gorria, 2021).

Preparatory Steps for Modifying the Live System within the Kubuntu 24.04 ISO

The initial phase of the customisation process involves making the contents of the existing Kubuntu 24.04 ISO accessible for modification. This is achieved by first creating a temporary directory, into which the ISO file is mounted using a loop device. Subsequently, an additional working directory is created, to which the read—only contents of the mounted ISO are copied in order to enable modification.

Following this step, the ISO image is unmounted and the original mount directory is removed. Entering the working directory containing the copied files, the *filesystem.squashfs* file located within the *casper* directory is extracted into a specified target directory using appropriate command parameters. Next, the required system directories are mounted, and access to the extracted squashfs environment is established via the *chroot* command, enabling direct modification of the live system filesystem.

System Configuration and Software Deployment

Upon entering the *squashfs—root* environment via *chroot*, the required software packages are installed, the system is updated, and localisation settings are configured in accordance with enterprise requirements. Figure 1 presents the list of applications included in the installation.

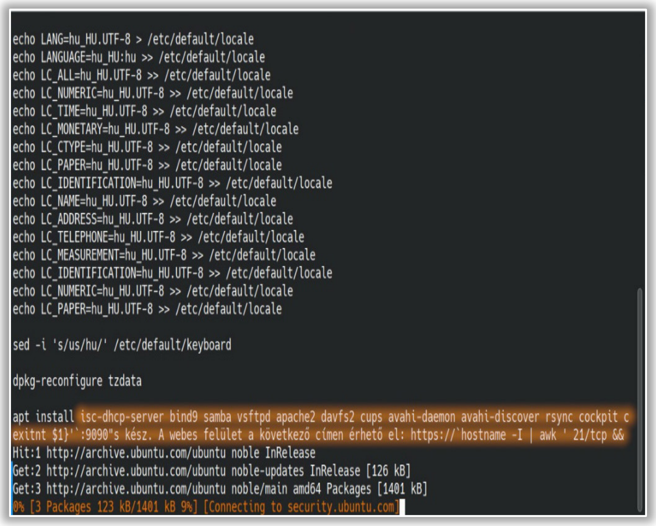


Figure 1: Localisation and configuration of the SME—oriented system

Following the completion of all modifications, the process proceeds with exiting the *chroot* environment and unmounting all previously mounted system directories. The original *filesystem.squashfs* file is then removed.

RESULTS

Commands Required for SME ISO Imaging

To facilitate the ISO imaging process for small– and medium–sized enterprises (SMEs), the following commands are employed to install the requisite software tools:

```
sudo apt install xorriso —y
sudo apt install isolinux —y
```

The first command installs the xorriso utility, a critical tool for constructing ISO 9660 file systems. The second command deploys the isolinux package, which is essential for bootstrapping the Linux kernel.

Commands for Preparing Modifications to the Live System within the Kubuntu 24.04 ISO

The sequence of commands encompasses the creation of a temporary directory for mounting the ISO contents, establishing a persistent directory for file operations, unmounting the ISO, deleting the temporary directory, extracting the SquashFS filesystem within the persistent directory, mounting essential system components required for chroot, and ultimately entering the ISO filesystem via chroot:

```
mkdir iso_contents
sudo mount —o loop kubuntu—24.04.2—desktop—amd64.iso iso_contents
mkdir custom_iso
sudo cp —r iso_contents/* custom_iso
sudo umount iso_contents
rm —r iso_contents
cd custom_iso
sudo unsquashfs —d squashfs—root casper/filesystem.squashfs
sudo mount ——bind /dev squashfs—root/dev
sudo mount ——bind /dev/pts squashfs—root/dev/pts
sudo mount ——bind /proc squashfs—root/proc
sudo mount ——bind /sys squashfs—root/sys
sudo mount ——bind /run squashfs—root/run
sudo chroot squashfs—root
```

Issues Arising from Fresh Kernel Installation

Before commencing system configuration, it is critical to highlight a particular challenge observed during customization. Installing a yet–to–be–deployed kernel within Kubuntu, or any distribution leveraging the KDE desktop environment, can trigger multiple functional issues. Specifically, on a machine running the live system, the wireless network interface remains inactive, while internet connectivity is only available via the Ethernet port. If the machine lacks an Ethernet interface, it becomes entirely incapable of establishing an internet connection. In this state, USB ports provide power to connected devices—such as illuminating a wired mouse—without enabling functional communication. Consequently, even a USB–to–Ethernet adapter fails to facilitate network access due to the absence of active data communication. The root cause lies in the kernel installed during the customization process, which lacks proper support for the system's diverse hardware components.

Notably, this issue manifests exclusively under the KDE desktop environment. In contrast, testing revealed that creating a Linux Mint 22.1 installer does not produce this problem: the kernel can be installed seamlessly, as the Cinnamon interface appears inherently immune to this hardware compatibility anomaly.

Commands Required for Localization and Configuration of SME Systems

The localization and installation of essential services and applications tailored for small– and medium–sized enterprises (SMEs) are executed using the following command.

— Updating package repositories:

```
apt update
```

— Performing a comprehensive system upgrade:

```
apt full—upgrade —y
```

— Applying localizations to the target language (Hungarian in this example)

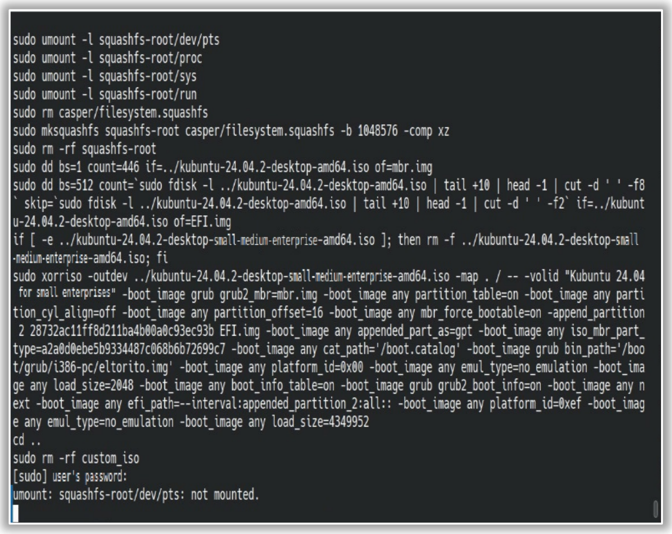
```
apt install language—pack—hu* language—pack—kde—hu hunspell—hu hyphen—hu —y
echo LANG=hu_HU.UTF—8 > /etc/default/locale
echo LANGUAGE=hu_HU:hu >> /etc/default/locale
echo LC_ALL=hu_HU.UTF—8 >> /etc/default/locale
sed —i 's/us/hu/' /etc/default/keyboard
dpkg—reconfigure tzdata
```

— Installing and enabling essential services:

```
apt install isc—dhcp—server bind9 samba vsftpd apache2 davfs2 cups avahi—daemon avahi—discover rsync cockpit cockpit—networkmanager fail2ban ufw wireguard openssh—server —y
systemctl enable isc—dhcp—server
systemctl enable bind9
systemctl enable smbd
systemctl enable vsftpd
systemctl enable apache2
a2enmod dav dav_fs && systemctl restart apache2
systemctl enable cups
systemctl enable avahi—daemon
systemctl enable cockpit
systemctl enable fail2ban
ufw allow 22/tcp && ufw allow 80,443/tcp && ufw allow 139,445/tcp && ufw allow 631/tcp && ufw allow 21/tcp && ufw allow 5353/udp && ufw ——force enable
echo "[✓] Installation complete. The web interface is available at the following address: https://`hostname —I | awk '{print $1}'`:9090"
exit
```

Commands Required for Finalizing the Installer

To finalize the installer, the commands depicted in Figure 2 must be executed.



```
sudo umount -l squashfs-root/dev/pts
sudo umount -l squashfs-root/proc
sudo umount -l squashfs-root/sys
sudo umount -l squashfs-root/run
sudo rm casper/filesystem.squashfs
sudo mksquashfs squashfs-root casper/filesystem.squashfs -b 1048576 -comp xz
sudo rm -rf squashfs-root
sudo dd bs=1 count=446 if=../kubuntu-24.04.2-desktop-amd64.iso of=mbr.img
sudo dd bs=512 count=`sudo fdisk -l ../kubuntu-24.04.2-desktop-amd64.iso | tail +10 | head -1 | cut -d ' ' -f8
` skip=`sudo fdisk -l ../kubuntu-24.04.2-desktop-amd64.iso | tail +10 | head -1 | cut -d ' ' -f2` if=../kubunt
u-24.04.2-desktop-amd64.iso of=EFI.img
if [ -e ../kubuntu-24.04.2-desktop-small-medium-enterprise-amd64.iso ]; then rm -f ../kubuntu-24.04.2-desktop-small
-medium-enterprise-amd64.iso; fi
sudo xorriso -outdev ../kubuntu-24.04.2-desktop-small-medium-enterprise-amd64.iso -map . / -- -volid "Kubuntu 24.04
 for small enterprises" -boot_image grub grub2_mbr=mbr.img -boot_image any partition_table=on -boot_image any parti
tion_cyl_align=off -boot_image any partition_offset=16 -boot_image any mbr_force_bootable=on -append_partition
 2 28732ac11ff8d211ba4b00a0c93ec93b EFI.img -boot_image any appended_part_as=gpt -boot_image any iso_mbr_part_
type=a2a0d0ebe5b9334487c068b6b72699c7 -boot_image any cat_path='/boot.catalog' -boot_image grub bin_path='/boo
t/grub/i386-pc/eltorito.img' -boot_image any platform_id=0x00 -boot_image any emul_type=no_emulation -boot_ima
ge any load_size=2048 -boot_image any boot_info_table=on -boot_image grub grub2_boot_info=on -boot_image any n
ext -boot_image any efi_path=--interval:appended_partition_2:all:: -boot_image any platform_id=0xef -boot_imag
e any emul_type=no_emulation -boot_image any load_size=4349952
cd ..
sudo rm -rf custom_iso
[sudo] user's password:
umount: squashfs-root/dev/pts: not mounted.
```

Figure 2: Commands necessary for the completion of the SME ISO

Before constructing the ISO image, it is essential to extract certain critical information and files from the source image intended for customization. This includes the boot sector image, the UEFI boot filesystem, and the associated structural metadata. In modern Ubuntu distributions, both the MBR boot code and the EFI image are not provided as standalone files; instead, they must be retrieved directly from the existing ISO image. Specifically, one resides within the boot sector, while the other is located on a dedicated partition. The commands required to perform these two extraction operations are as follows:

sudo dd bs=1 count=446 if=../kubuntu—24.04.2—desktop—amd64.iso of=mbr.img
sudo dd bs=512 count=`sudo fdisk —l ../kubuntu—24.04.2—desktop—amd64.iso | tail +10 | head —1 | cut —d ' ' —f8` skip=`sudo fdisk —l ../kubuntu—24.04.2—desktop—amd64.iso | tail +10 | head —1 | cut —d ' ' —f2` if=../kubuntu—24.04.2—desktop—amd64.iso of=EFI.img

To verify the success of the customization process, the system contained within the ISO image must be booted, allowing for the visualization of the applied modifications. The overall outcome of the process is illustrated in Figure 3.

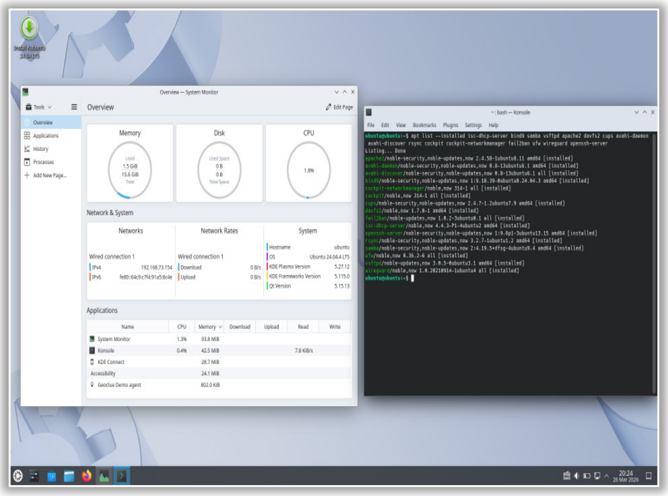


Figure 3: The ready—to—use, fully installed system

CONCLUSIONS

The findings of this study demonstrate that the customization of Linux—based operating systems and the development of bespoke installer images hold substantial strategic significance for the IT operations of small— and medium—sized enterprises (SMEs). The research confirmed that structured, reproducible modification of an existing ISO image—through management of the SquashFS filesystem, utilization of a chroot environment, and subsequent ISO regeneration—can be executed efficiently. This methodology enables organizations to create pre—configured systems tailored to their specific operational requirements.

For SMEs, the optimized use of resources is particularly critical, and customized Linux ISOs contribute significantly to this objective. Deploying pre—configured systems reduces installation and configuration time, minimizes the risk of human error, and simplifies the management of IT infrastructure. Consequently, even smaller organizations can implement professional, consistent, and easily maintainable systems without necessitating substantial additional resources.

The investigation also highlighted that compatibility issues—especially in the domains of kernel and driver management—can impose disproportionate burdens on SMEs. Customized installation solutions, however, provide a mechanism for addressing these challenges proactively, ensuring a stable and fully operational environment post—deployment. This not only enhances system reliability but also lowers ongoing maintenance costs.

Our results indicate that customization represents not merely a technological advantage but a strategic asset for SMEs. It enables the creation of scalable and automatable IT environments that support organizational growth and operational efficiency. This is particularly crucial for organizations lacking a dedicated, large—scale IT team.

The study validates that the adoption of customized Linux ISOs can confer a substantial competitive advantage for SMEs. The methodology presented facilitates faster deployment, standardization, and long—term stability of IT systems. Future development directions include deeper integration of automated installation workflows and the design of solutions that further reduce operational overhead within the SME sector.

References

[1] Abbott, D. (2018). Installing Linux. Linux for Embedded and Real—Time Applications 15—28.
[2] Alawadh, O., Abdulrahman, R., Alkhaldi, F., Alsheddi, K., Buaysha, M., Rashidi, H. (2021). Virtualization Technologies: Overview, Differences &

Similarities, International Journal of Computer Trends and Technology (IJCTT), 69(10), 39—42.

[3] Aljabari, G. (2012). Virtualization of IT Infrastructure for Small and Medium Businesses. The 2nd IEEE International Conference on Communications and Information Technology (ICCIT) IEEE https://scholar.ppu.edu/server/api/core/bitstreams/ 4686d7c9—79ae—4cd3—afd5—bb5a48d69ecd /content

[4] Cashwell, B. (2024). Small Business Virtualization: Improving Operations and Reducing Costs. https://www.veeam.com/blog/small—business—virtualization—guide.html

[5] Dempster, J. (2001). The Personal Computer. The Laboratory Computer. 12—44.

[6] Díaz, G., Rojas, P., Barrios, C. (2019). Methodology for Tailored Linux Distributions Development for HPC Embedded Systems. Communications in Computer and Information Science High Performance Computing pp. 280—290.

[7] Ganney, P.S., Pisharody S., Claridge. E. (2020). Software engineering. Clinical Engineering. 131—168.

[8] Gediya, J., Singh, J., Kushwaha P., Srivastava R., Wang Z. (2019). Open—Source Software. Software Engineering for Embedded Systems. 207—244.

[9] Koutoupis, P. (2018). DIY: Build a Custom Minimal Linux Distribution from Source. Linux Journal. https://www.linuxjournal.com/content/diy—build—custom—minimal—linux—distribution—source

[10] Linux Installation. Computer Science https://www.sciencedirect.com/topics/computer—science /linux—installation

[11] Polstra, P. (2015). Chapter 3 — Installing a base operating system. Hacking and Penetration Testing with Low Power Devices. 27—54.

[12] Research Gate. Current_Approaches_and_Future_Opportunities , https://www.researchgate.net /publication/228918419 Linux_and_Real—Time_

[13] Sahoo, J., Mohapatra, S., Lath, R. (2021). Virtualization Technologies: Overview, Differences & Similarities. International Journal of Computer Trends and Technology. 69(10), 39—42.

[14] Santana, M. (2013). Eliminating the Security Weakness of Linux and Unix Operating Systems. Computer and Information Security Handbook. 183—196.

[15] Scordino, C., Lipari, G. (2006). Linux and Real—Time: Current Approaches and Future Opportunities. https://www.researchgate.net/publication/228918419 Linuxand_Real—Time_Current Approaches_and_Future_Opportunities

[16] Sinclair, I. (2011). Computer Construction and Software. Electronics Simplified. 281—295.

[17] Yadav, P.R. (2024). The LINUX Operating System: An Introduction. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), 26(2), 424—432.

[18] Whittaker, G. (2025). How to Build Custom Distributions from Scratch. https://www.linuxjournal.com/content/how—build—custom—distributions—scratch

ISSN: 2067–3809