GlobalLogic
A Hitachi Group Company

EDUCATION

Smart Start: Linux/Networking
Starting with Linux

Sergii Kudriavtsev

Agenda

1. GNU/Linux overview
2. Linux Drivers, GUI, Userspace Processes
3. Linux Distributives
4. Users, Groups, Passwords
5. File Attributes
6. Processes vs Daemons
7. Background Processes
8. Signals and Killing
9. Ubuntu installation to virtual machine

GNU/Linux Overview

GNU + Linux => GNU/Linux

- **GNU** is an extensive collection of free software (385 packages as of September 2023), which can be used as an operating system or can be used in parts with other operating systems. The use of the completed GNU tools led to the family of operating systems popularly known as Linux. Most of GNU is licensed under the GNU Project's own General Public License (GPL).
- An operating system built using the Linux kernel and GNU tools and utilities is considered a variant of GNU, and promotes the term *GNU/Linux* for such systems.
- Linux kernel was originally written in 1991 by Linus Torvalds for his i386-based PC, and it was soon adopted as the kernel for the GNU operating system, which was written to be a free replacement for Unix.
- The Linux kernel supports various hardware architectures, providing a common platform for software, including proprietary software.

GlobalLogic
A Hitachi Group Company
EDUCATION

Copyright, GNU licenses

- For the development of needed software, a license called the GNU General Public License was written, with the goal to guarantee users freedom to share and change free software.
- In 1991, the GNU Lesser General Public License (LGPL), then known as the Library General Public License, was written for the GNU C Library to allow it to be linked with proprietary software.

- **What is an Open Source License?**
- Open source software licenses govern how others – besides the originator – can use, modify, or distribute software code.
- They grant other users the permission and rights to use or repurpose the code for new applications or to include the code in other projects.
- One of the main advantages of open source code is its visibility, which makes it easier to troubleshoot problems and to understand better how something works when the documentation is either lacking or incorrect.
- Depending on the type of open source license, you may even be allowed to modify the original source code to tailor it to your needs or fix any issues you find.
- The license will determine whether this is possible, and under what terms. For example, you may be required to make any modifications publicly available.

GlobalLogic
A Hitachi Group Company
EDUCATION

Copyright, GNU licenses

- **What are the different open source licenses?**
- There are over 80 variations of open-source licenses, but they generally fall into one of two primary categories: copyleft and permissive:

- **Copyleft license** is a license type in which code derived from the original open source code inherits its license terms: **GNU General Public License (GPL(**GPLv2 and GPLv3**)), Affero GPL (AGPL), Lesser General Public License (LGPL) , Eclipse Public License (EPL), Mozilla Public License (MPL).**

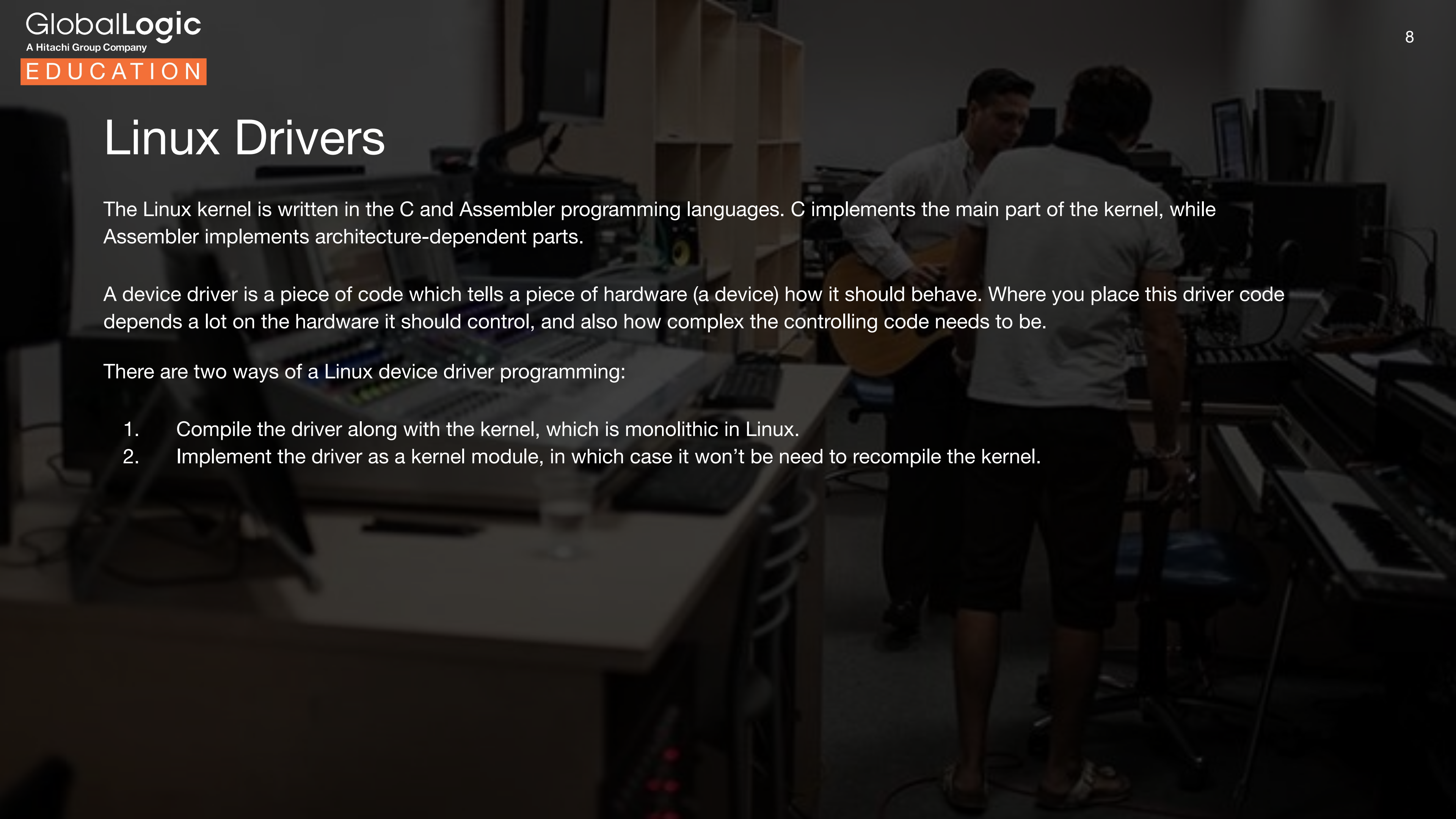
- **Permissive license** is a license type that provides more freedom for reuse, modification, and distribution: **Apache License, MIT License, Berkeley Source Distribution (BSD) License**

- **Linux** is provided under the GNU General Public License version 2 only, but it contains files under other compatible licenses.

GlobalLogic
A Hitachi Group Company
EDUCATION

Linux Drivers, GUI, Userspace Processes

GlobalLogic
A Hitachi Group Company
EDUCATION



Linux Drivers

The Linux kernel is written in the C and Assembler programming languages. C implements the main part of the kernel, while Assembler implements architecture-dependent parts.

A device driver is a piece of code which tells a piece of hardware (a device) how it should behave. Where you place this driver code depends a lot on the hardware it should control, and also how complex the controlling code needs to be.

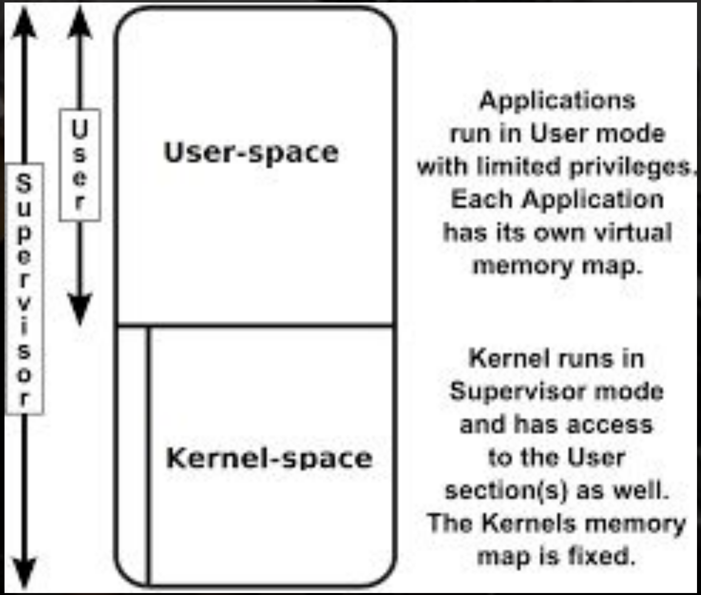
There are two ways of a Linux device driver programming:

1. Compile the driver along with the kernel, which is monolithic in Linux.
2. Implement the driver as a kernel module, in which case it won’t be need to recompile the kernel.

Linux Drivers

In Linux, device drivers are typically divided into two categories: user-space and kernel-space.User-space device drivers are written in a high-level language like C or C++ and run in the user-space of the operating system. They interact with the kernel through system calls and do not have direct access to hardware resources. These drivers are typically simpler and easier to develop than kernel-space drivers, but they are also slower and less reliable due to the overhead of system calls and user-kernel context switches. Examples of user-space drivers include libusb, libieee1284, and libftdi.

Kernel-space device drivers are written in C and run in the kernel-space of the operating system. They have direct access to hardware resources and can execute without the overhead of system calls and user-kernel context switches. These drivers are typically more complex and harder to develop than user-space drivers, but they are also faster and more reliable. Examples of kernel-space drivers include network drivers for managing network interfaces, sound drivers for audio devices, and display drivers for graphics cards. Also the Linux USB, IEEE 1284, and FTDI drivers.It's worth noting that both types of drivers can coexist in the same system and can interact with each other through appropriate interfaces, such as the kernel's sysfs or the user-space libudev library.

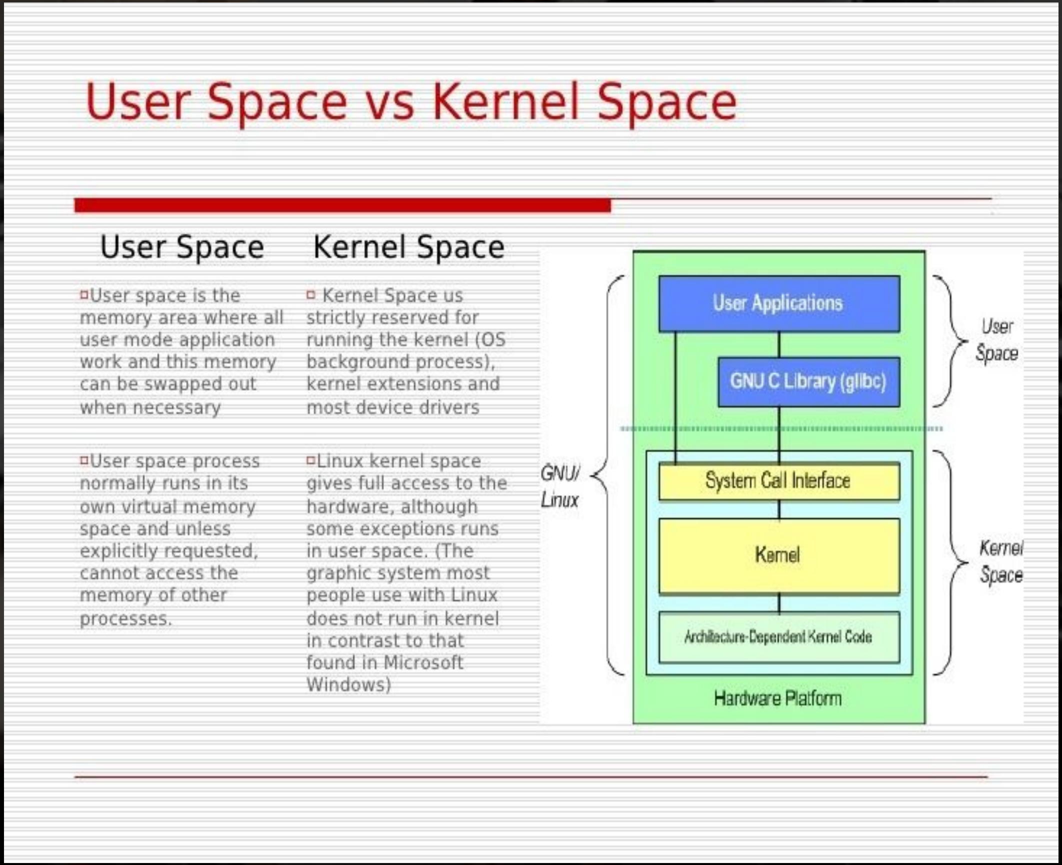

Supervisor
User
User-space
Kernel-space
Applications run in User mode with limited privileges. Each Application has its own virtual memory map.
Kernel runs in Supervisor mode and has access to the User section(s) as well. The Kernels memory map is fixed.

GlobalLogic
A Hitachi Group Company
EDUCATION

Linux Drivers

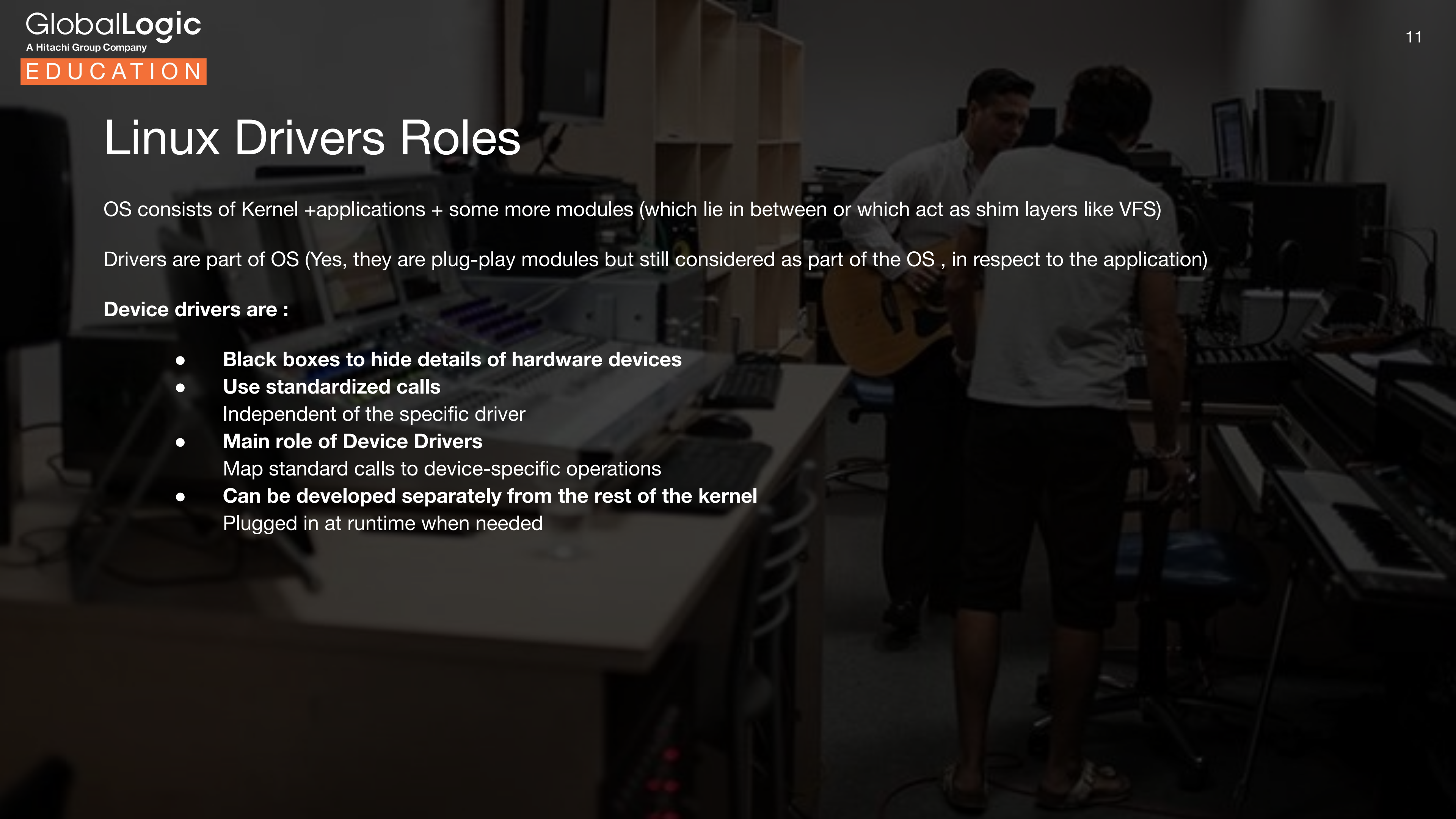
Some hardware, such as network cards, or video cards, typically require the driver to respond very quickly in order to ensure good performance. If these drivers are placed within the kernel (in kernel space), they will not be affected by things such as context switches, and time penalties related to copying memory to/from user-space. There are means to map memory in an efficient way for user-space drivers, but likely, performance will not be as high as when running in the kernel.

On the other hand, some hardware does not have the same requirements on the driver in terms of response time, an example could be a driver for SPI communication, or GPIO communication. The drivers still need to have access to the hardware somehow, and often very generic drivers are then created, allowing access to the hardware, but not specifying any application-specific behavior. The generic drivers are placed in the kernel, and can be re-used for many different user-space drivers, an example of this is the spidev driver.



User Space vs Kernel Space

User Space	Kernel Space
▫User space is the memory area where all user mode application work and this memory can be swapped out when necessary	▫ Kernel Space us strictly reserved for running the kernel (OS background process), kernel extensions and most device drivers
▫User space process normally runs in its own virtual memory space and unless explicitly requested, cannot access the memory of other processes.	▫Linux kernel space gives full access to the hardware, although some exceptions runs in user space. (The graphic system most people use with Linux does not run in kernel in contrast to that found in Microsoft Windows)



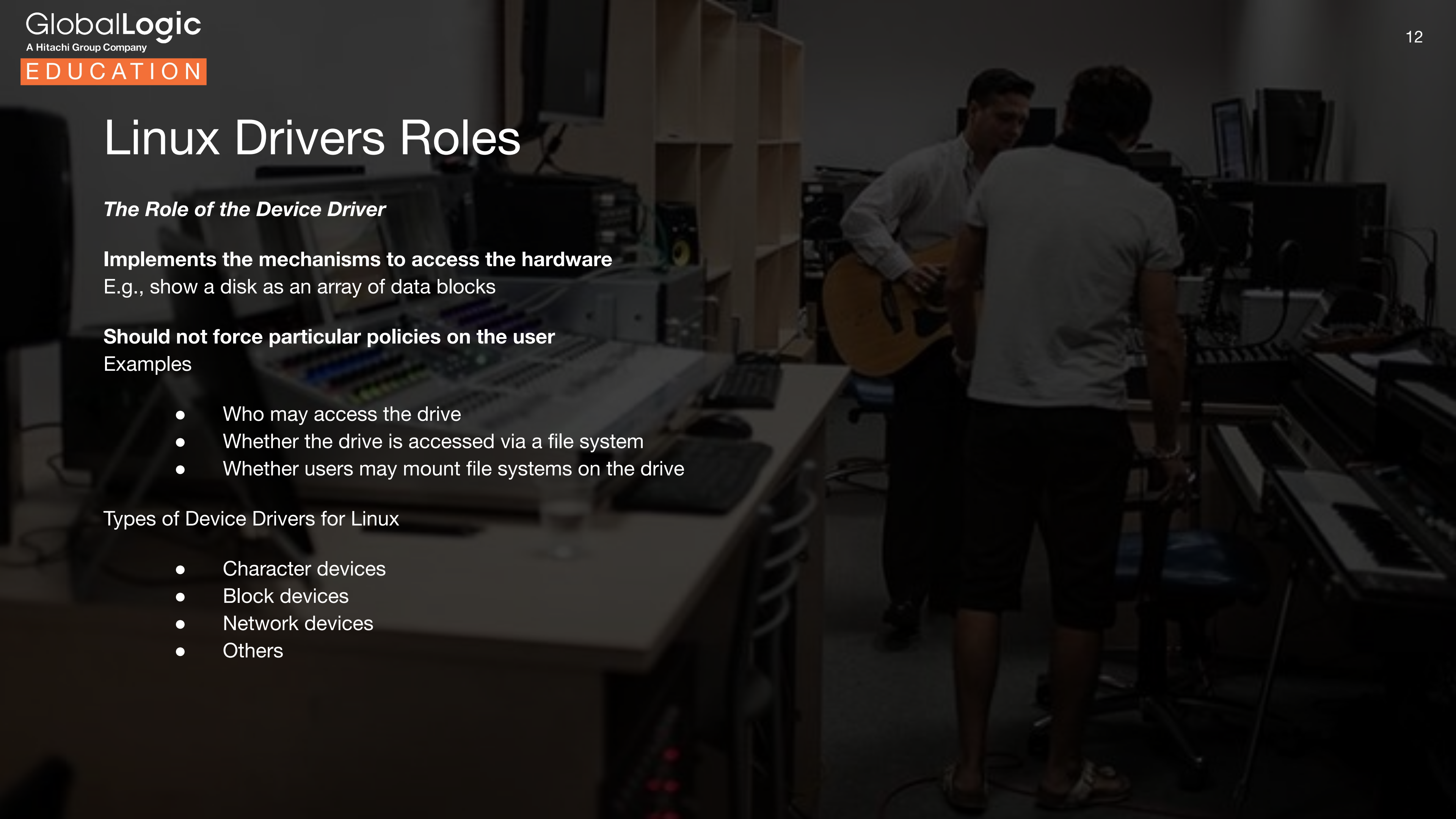
Linux Drivers Roles

OS consists of Kernel +applications + some more modules (which lie in between or which act as shim layers like VFS)

Drivers are part of OS (Yes, they are plug-play modules but still considered as part of the OS , in respect to the application)

Device drivers are :

- **Black boxes to hide details of hardware devices**
- **Use standardized calls**
  Independent of the specific driver
- **Main role of Device Drivers**
  Map standard calls to device-specific operations
- **Can be developed separately from the rest of the kernel**
  Plugged in at runtime when needed



Linux Drivers Roles

The Role of the Device Driver

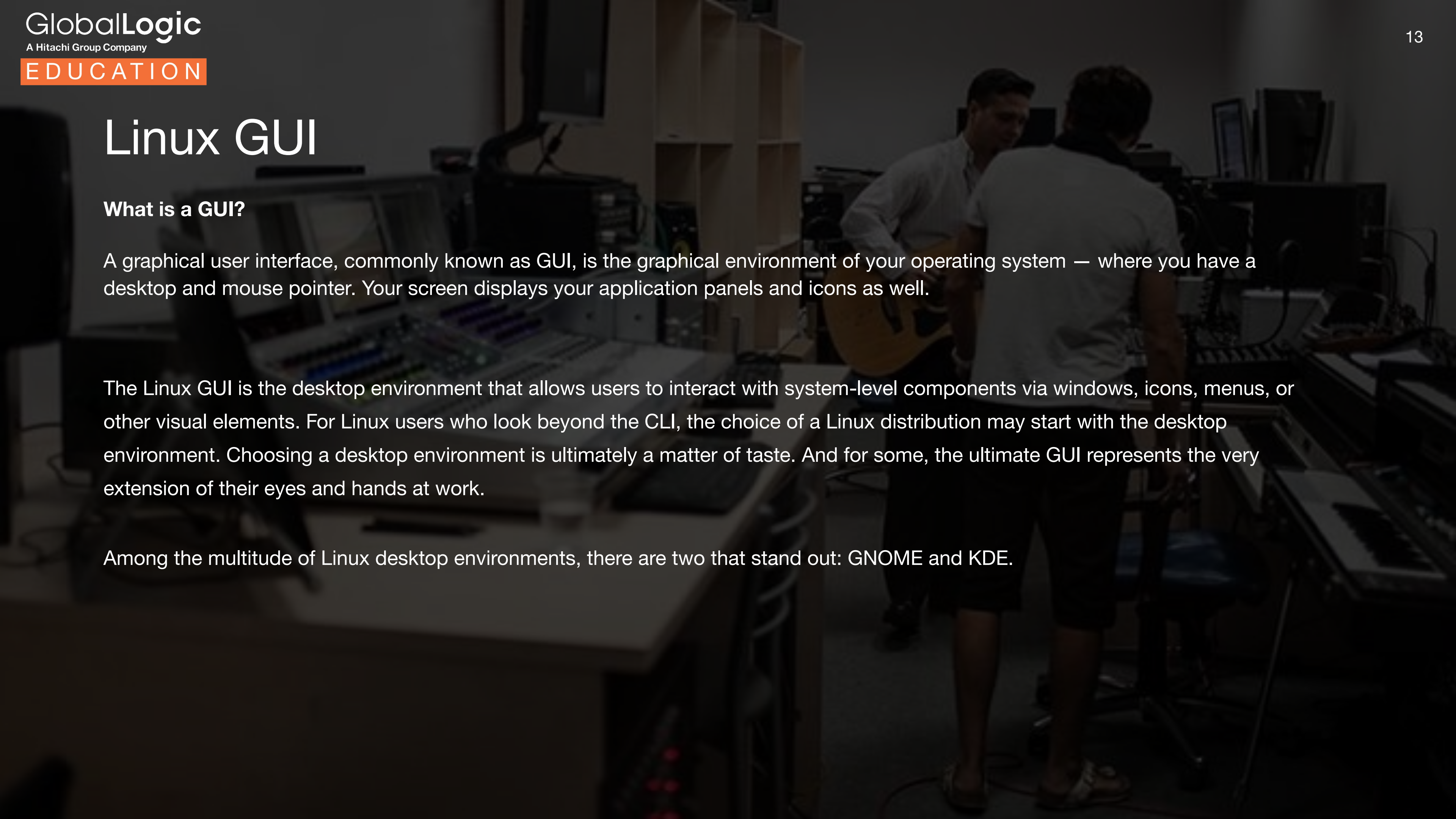
Implements the mechanisms to access the hardware
E.g., show a disk as an array of data blocks

Should not force particular policies on the user
Examples

- Who may access the drive
- Whether the drive is accessed via a file system
- Whether users may mount file systems on the drive

Types of Device Drivers for Linux

- Character devices
- Block devices
- Network devices
- Others



Linux GUI

What is a GUI?

A graphical user interface, commonly known as GUI, is the graphical environment of your operating system — where you have a desktop and mouse pointer. Your screen displays your application panels and icons as well.

The Linux GUI is the desktop environment that allows users to interact with system-level components via windows, icons, menus, or other visual elements. For Linux users who look beyond the CLI, the choice of a Linux distribution may start with the desktop environment. Choosing a desktop environment is ultimately a matter of taste. And for some, the ultimate GUI represents the very extension of their eyes and hands at work.

Among the multitude of Linux desktop environments, there are two that stand out: GNOME and KDE.

Linux GUI

GNOME
With its current GNOME 3 (or GNOME Shell) iteration, this GUI platform is one of the most common Linux desktop environments. Nowadays, almost every major Linux distribution comes with GNOME as the default GUI. The Linux open source community also created GNOME Extensions, which overcomes some of the infamous shortcomings of GNOME and extends the desktop functionality to suit a variety of needs. When it's not the default desktop environment (such as with Linux Mint's Cinnamon desktop), GNOME can easily be installed and adapted.

KDE
Linux administrators usually look for a desktop environment that is relatively easy to use, lightweight, and efficient. KDE combines all of these attributes into a reliable and speedy desktop interface. Users familiar with Windows (up to version 7) would feel very much at home with KDE.

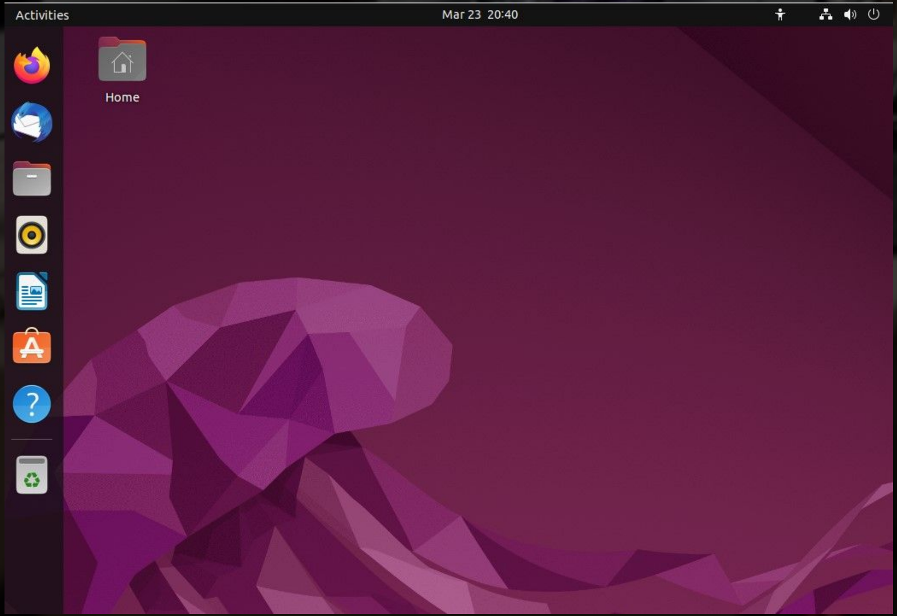
KDE has become a very robust desktop environment over the last few iterations, and versions of KDE have been released for almost every major Linux distribution.

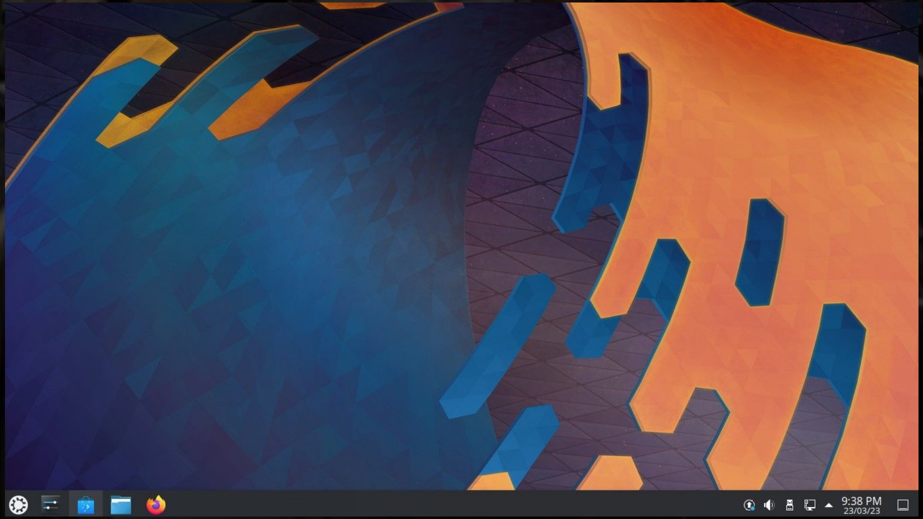
If there's one ideal desktop for Linux administrators, KDE comes very close.

GlobalLogic
A Hitachi Group Company
EDUCATION

Linux GUI

GNOME





Linux GUI

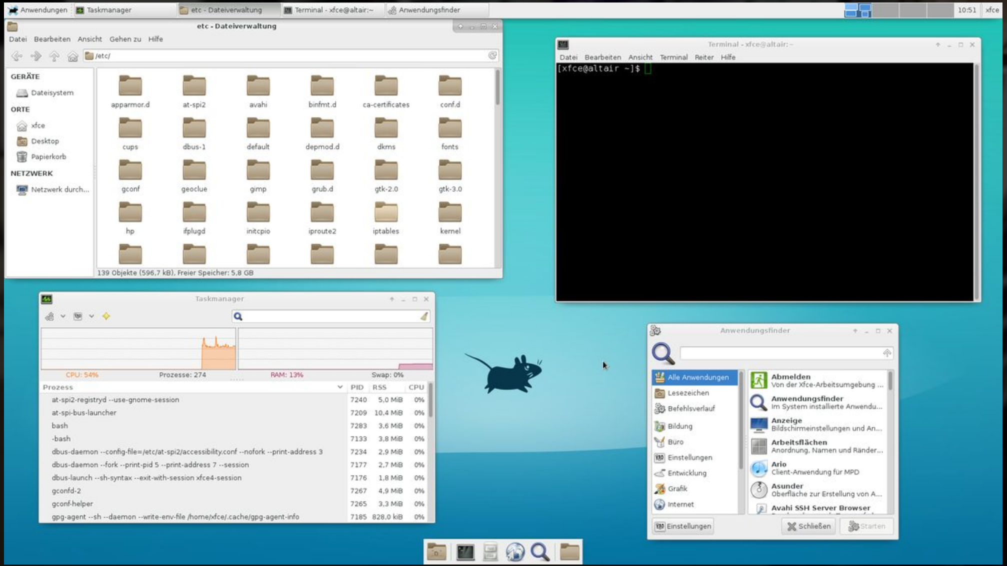
XFCE

The XFCE desktop environment is an excellent choice for those who want to have a more lightweight and customizable experience than GNOME offers. The interface can be customized, and the features that you use most are available with one click from your application dock or menu bar, so it's a good choice for PC enthusiasts that enjoy customizing their desktops.
XFCE proves to be a great choice if you need an environment that balances performance with user experience. It's light on resources but still provides powerful customizations and features.
XFCE is so lightweight that it runs surprisingly well on older hardware. The interface, which might seem familiar to Windows users, thanks to its layout which feels modern and is visually appealing despite being quite light on system resources.

LXDE

LXDE is another lightweight desktop environment that uses system resources sparingly, which means it can be used with a cheaper embedded board (like a Raspberry Pi) or an old salvaged computer.
The end result is that LXDE is a desktop environment that's lightweight and fast. It uses less RAM than most of the other desktop environments in our list, and it has fewer dependencies on different distributions or platforms.
LXDE provides the user with an easy-to-use interface that is responsive and simple to learn. If you're looking for a free, lightweight desktop environment that is easy to use and provides the basics of what a Linux interface needs, LXDE might be for you.
LXDE doesn't have as many features as some of the other environments do , but it also means that your system components can be a little on the older side when choosing a computer to run it from.

Linux GUI



139 Objekte (596,7 kB), Freier Speicher: 5,8 GB

```
[xfce@altair ~]$
```

```
CPU: 54%          Prozesse: 274          RAM: 13%          Swap: 0%
Prozess                                                                      PID   RSS        CPU
  at-spi2-registryd --use-gnome-session                                      7240  5,0 MiB    0%
  at-spi-bus-launcher                                                        7209  10,4 MiB   0%
  bash                                                                       7283  3,6 MiB    0%
  -bash                                                                      7133  3,8 MiB    0%
  dbus-daemon --config-file=/etc/at-spi2/accessibility.conf --nofork --print-address 3   7234  2,9 MiB    0%
  dbus-daemon --fork --print-pid 5 --print-address 7 --session               7177  2,7 MiB    0%
  dbus-launch --sh-syntax --exit-with-session xfce4-session                  7176  1,8 MiB    0%
  gconfd-2                                                                   7267  4,9 MiB    0%
  gconf-helper                                                               7265  3,3 MiB    0%
  gpg-agent --sh --daemon --write-env-file /home/xfce/.cache/gpg-agent-info  7185  828,0 kiB  0%
```

Abmelden
Von der Xfce-Arbeitsumgebung ...

Anwendungsfinder
Im System installierte Anwendu...

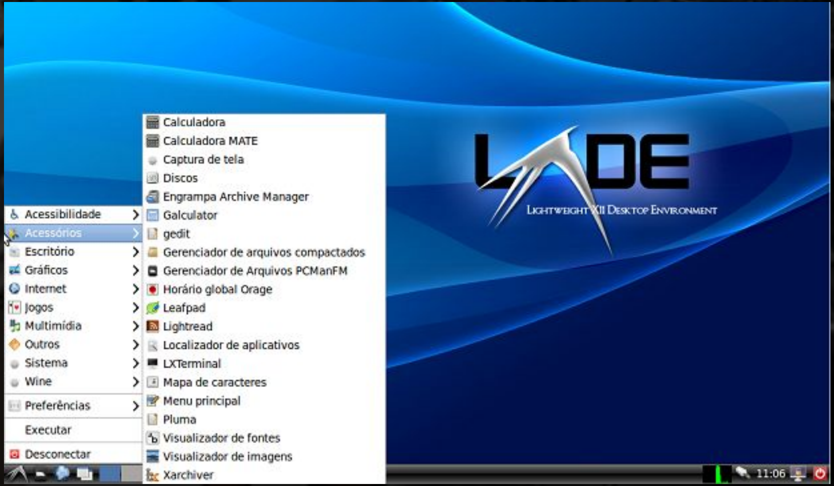
Anzeige
Bildschirmeinstellungen und An...

Arbeitsflächen
Anordnung, Namen und Ränder...

Ario
Client-Anwendung für MPD

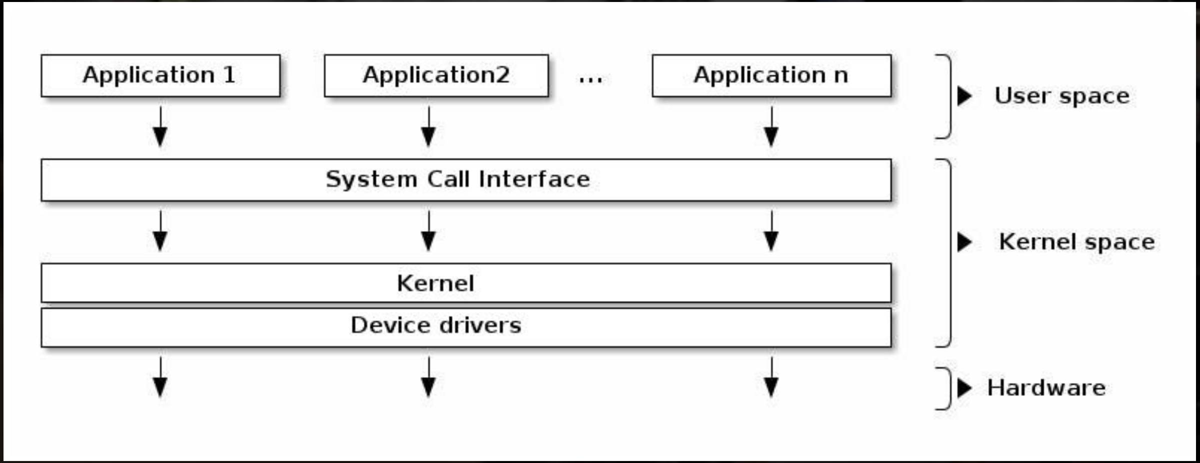
Asunder
Oberfläche zur Erstellung von A...

Avahi SSH Server Browser



Userspace Processes

There is a stack of services that underpin user-run processes in Linux. User-run processes rely on services provided by the kernel. The kernel is a special part of the operating system, which handles a variety of low-level operations in a privileged running mode (see the section on kernel space below).

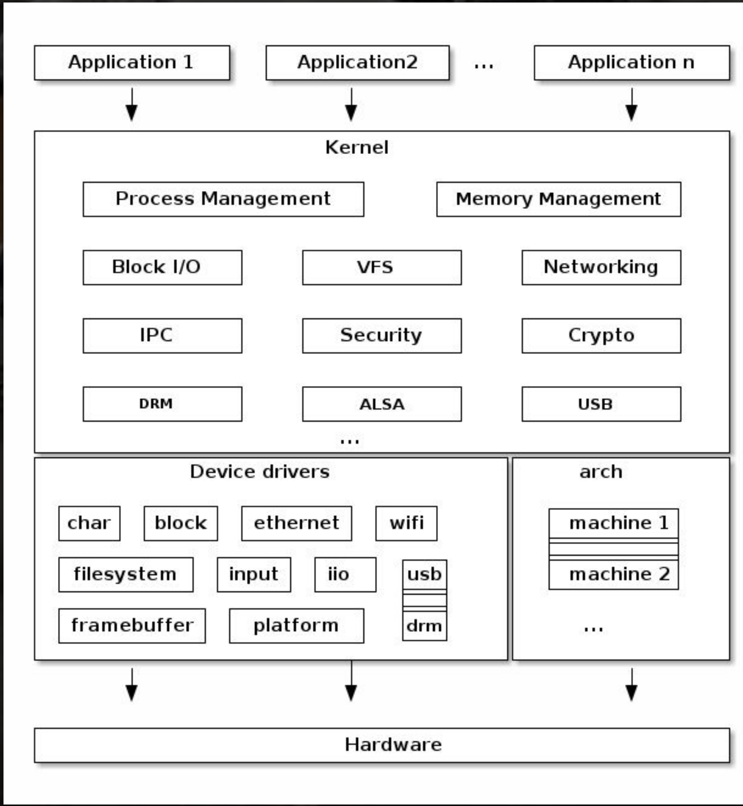


GlobalLogic
A Hitachi Group Company
EDUCATION

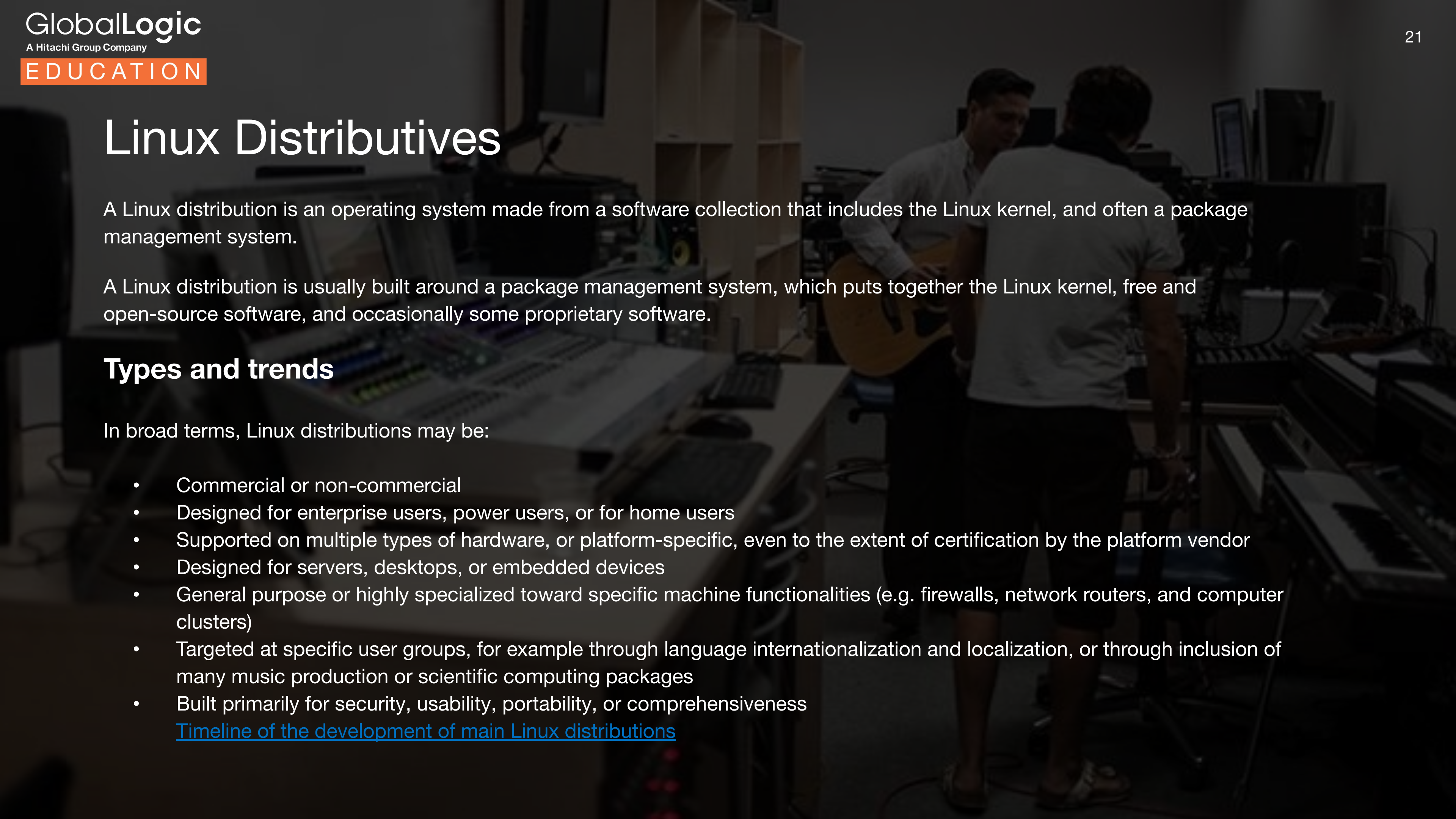
Userspace

A user space process is executed by a user in the operating system, rather than being part of the operating system itself. It might also be executed by an init system (e.g. systemd), but it isn't part of the kernel. User space is the area of memory that non-kernel applications run in. User space processes literally run in the user space part of memory. A user space process runs in user mode, which is the non-privileged execution mode that the process' instructions are executed with. User mode processes have to switch to kernel mode when they want to consume services provided by the kernel (e.g. disk I/O, network access). Switching to kernel mode involves triggering a syscall to be executed by the kernel.

User mode execution of user-run processes ensures that a user space process cannot access or modify memory managed by the kernel, and can't interfere with another process' execution. This is an important security control in ensuring that user-run processes cannot corrupt or interfere with the operating system.



Linux Distributives



Linux Distributives

A Linux distribution is an operating system made from a software collection that includes the Linux kernel, and often a package management system.

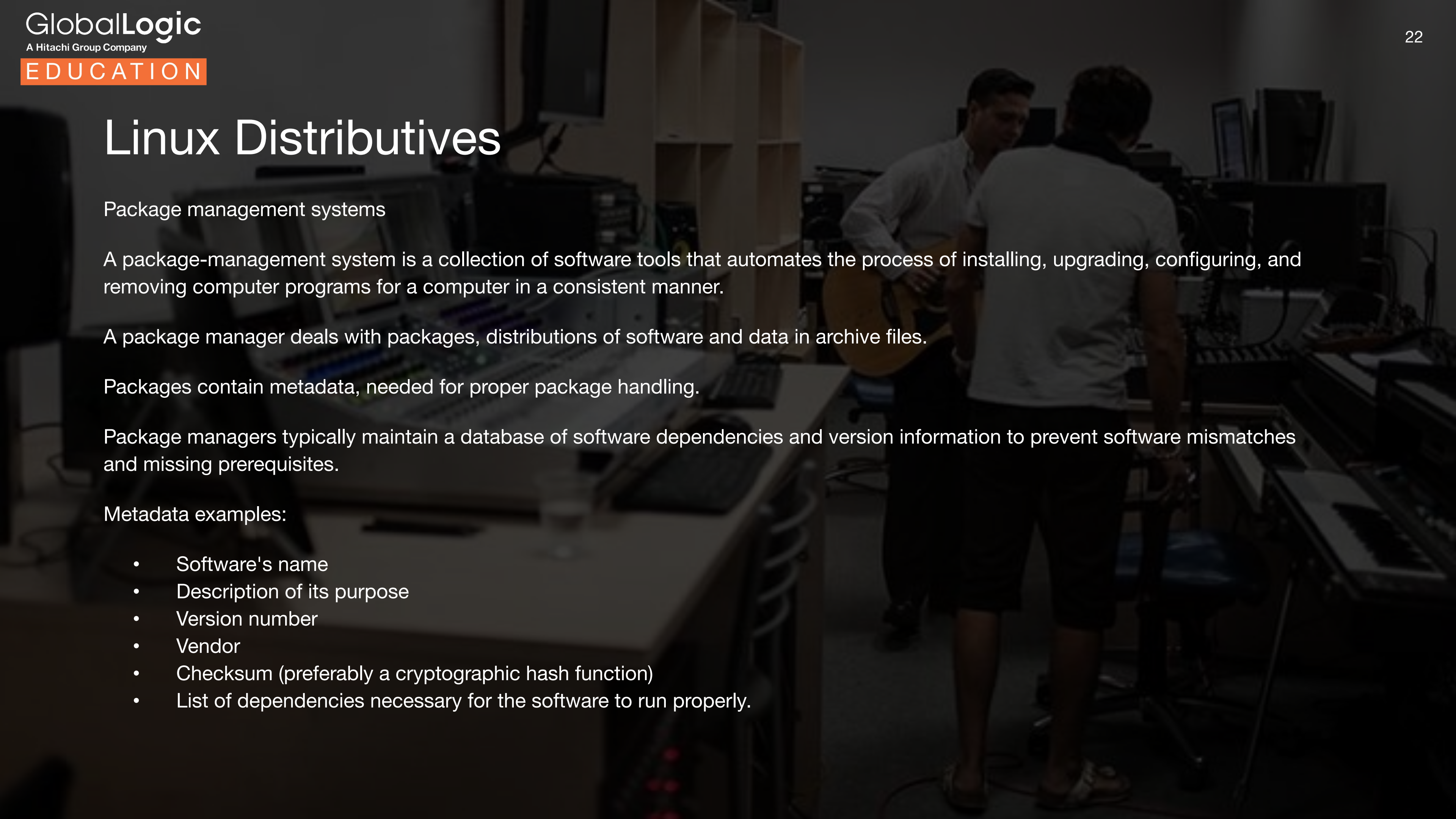
A Linux distribution is usually built around a package management system, which puts together the Linux kernel, free and open-source software, and occasionally some proprietary software.

Types and trends

In broad terms, Linux distributions may be:

- Commercial or non-commercial
- Designed for enterprise users, power users, or for home users
- Supported on multiple types of hardware, or platform-specific, even to the extent of certification by the platform vendor
- Designed for servers, desktops, or embedded devices
- General purpose or highly specialized toward specific machine functionalities (e.g. firewalls, network routers, and computer clusters)
- Targeted at specific user groups, for example through language internationalization and localization, or through inclusion of many music production or scientific computing packages
- Built primarily for security, usability, portability, or comprehensiveness

  [Timeline of the development of main Linux distributions]



Linux Distributives

Package management systems

A package-management system is a collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs for a computer in a consistent manner.

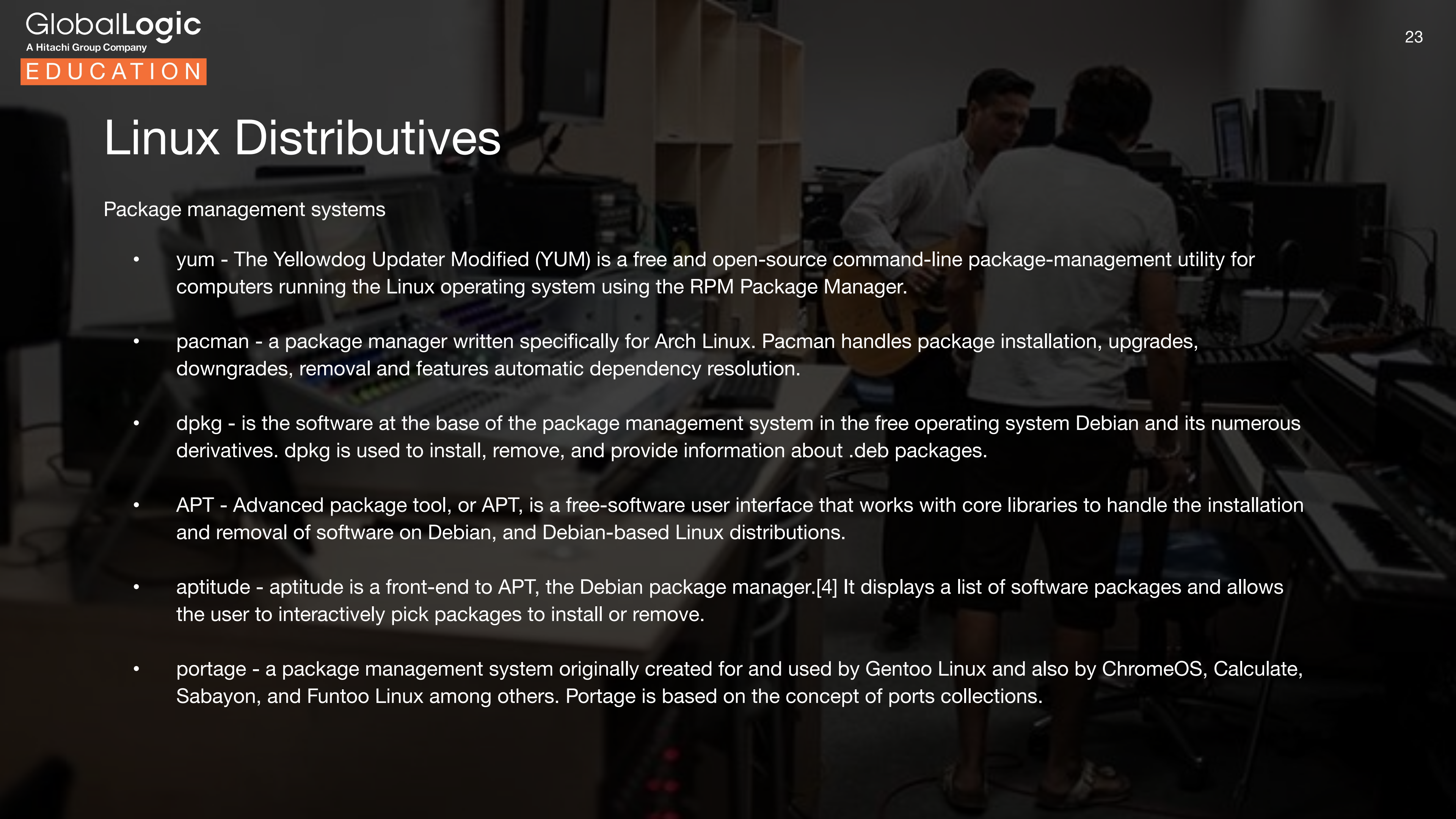
A package manager deals with packages, distributions of software and data in archive files.

Packages contain metadata, needed for proper package handling.

Package managers typically maintain a database of software dependencies and version information to prevent software mismatches and missing prerequisites.

Metadata examples:

- Software's name
- Description of its purpose
- Version number
- Vendor
- Checksum (preferably a cryptographic hash function)
- List of dependencies necessary for the software to run properly.

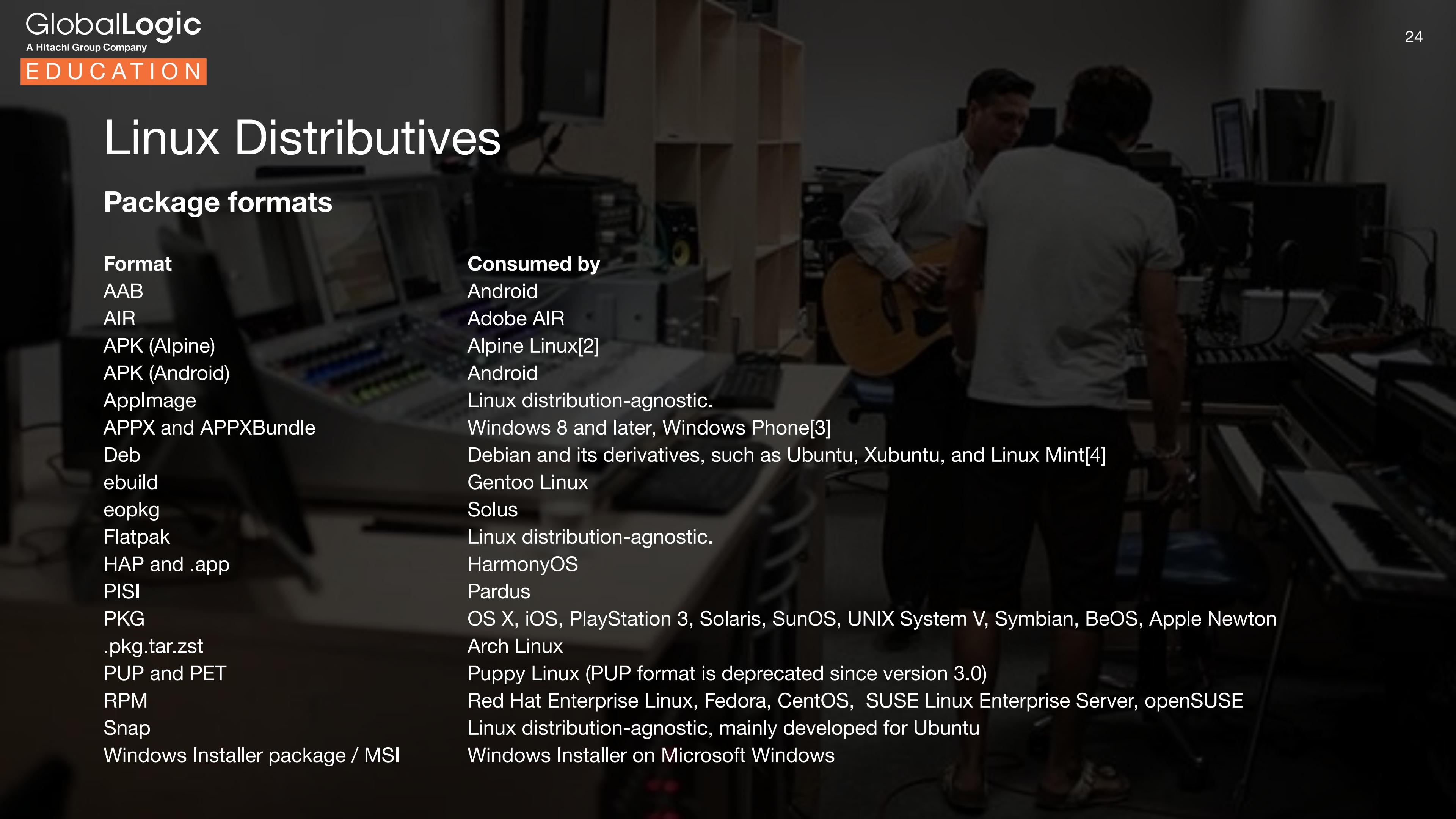


Linux Distributives

Package management systems

- yum - The Yellowdog Updater Modified (YUM) is a free and open-source command-line package-management utility for computers running the Linux operating system using the RPM Package Manager.
- pacman - a package manager written specifically for Arch Linux. Pacman handles package installation, upgrades, downgrades, removal and features automatic dependency resolution.
- dpkg - is the software at the base of the package management system in the free operating system Debian and its numerous derivatives. dpkg is used to install, remove, and provide information about .deb packages.
- APT - Advanced package tool, or APT, is a free-software user interface that works with core libraries to handle the installation and removal of software on Debian, and Debian-based Linux distributions.
- aptitude - aptitude is a front-end to APT, the Debian package manager.[4] It displays a list of software packages and allows the user to interactively pick packages to install or remove.
- portage - a package management system originally created for and used by Gentoo Linux and also by ChromeOS, Calculate, Sabayon, and Funtoo Linux among others. Portage is based on the concept of ports collections.

GlobalLogic
A Hitachi Group Company
EDUCATION

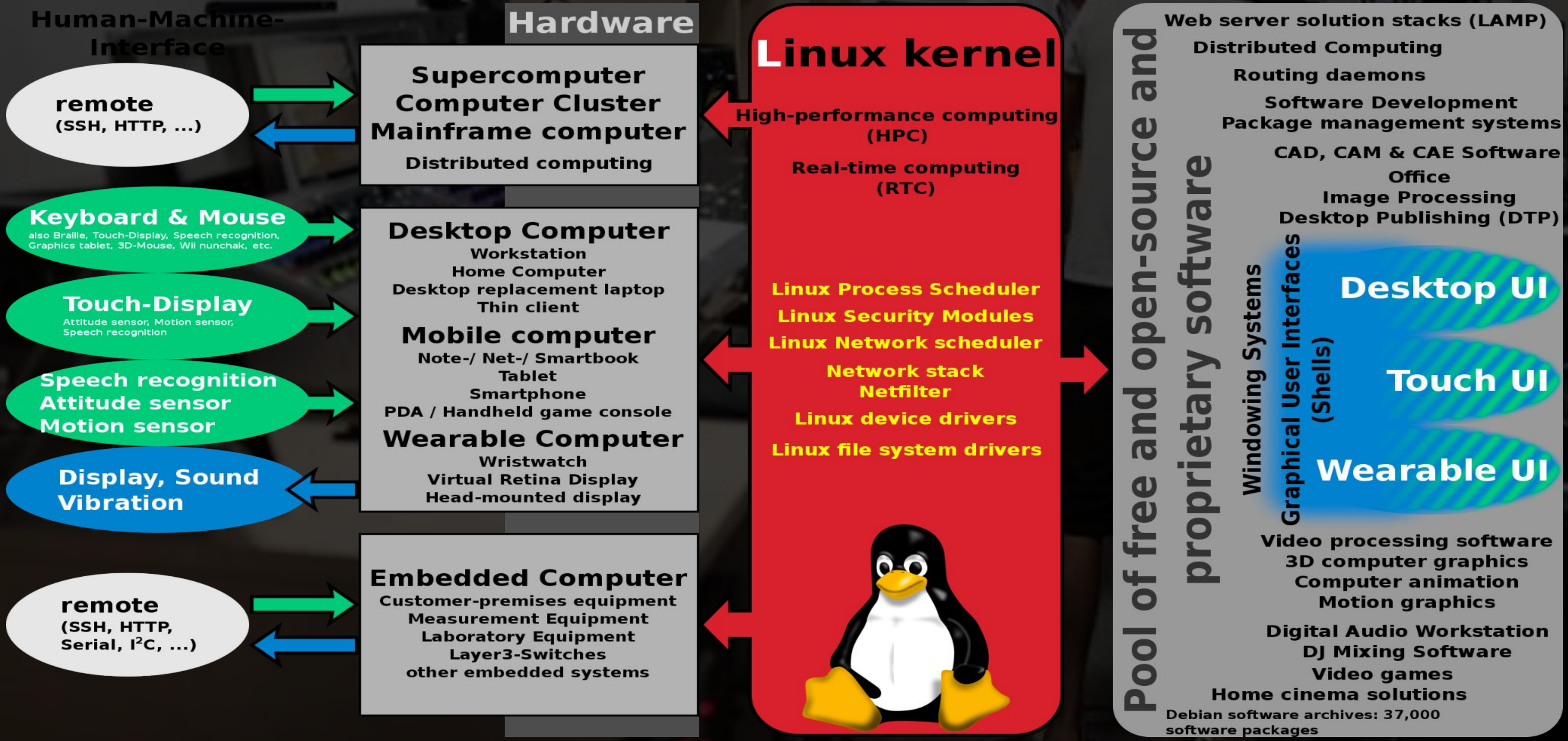


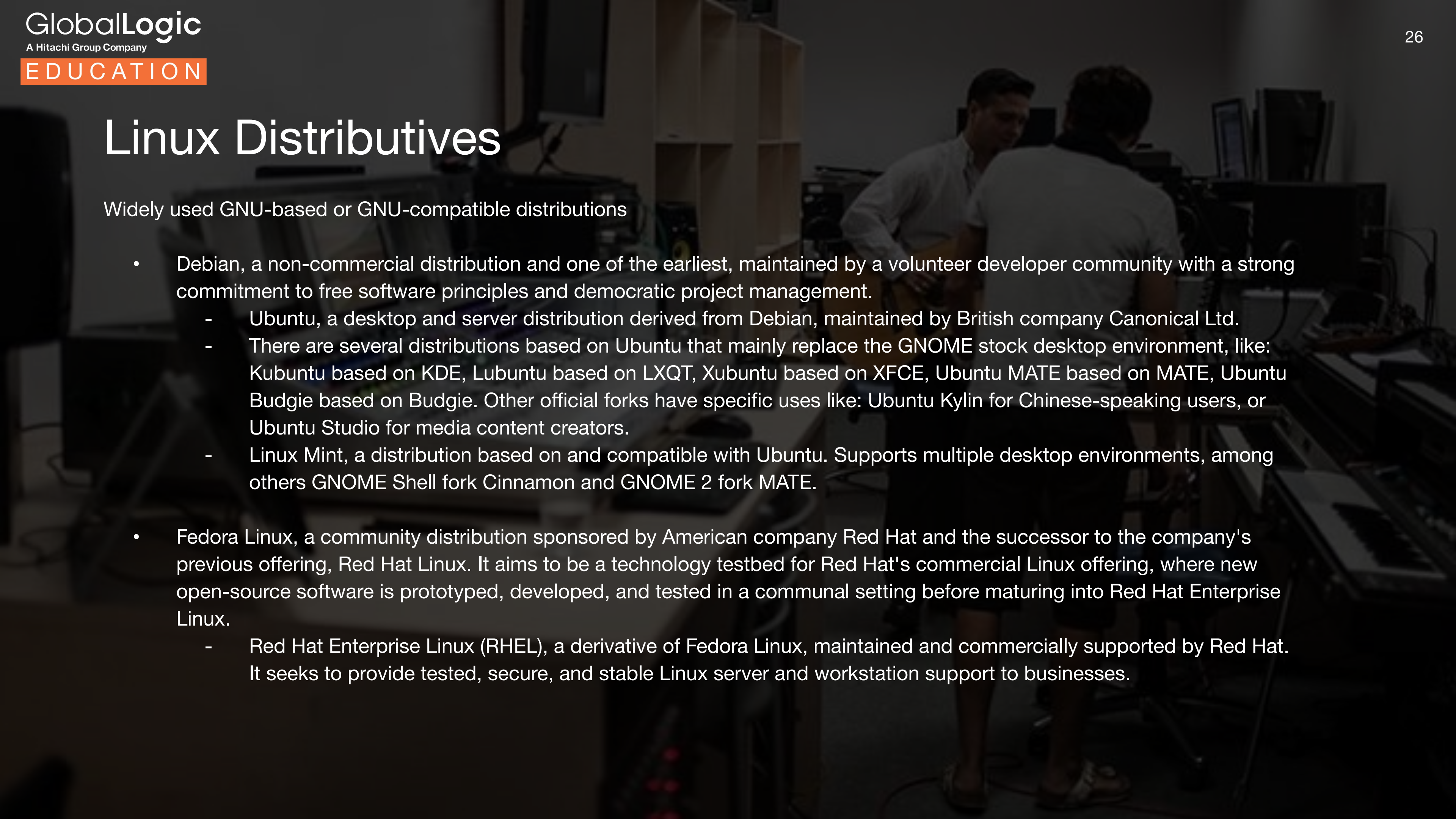
Linux Distributives

Package formats

Format	Consumed by
AAB	Android
AIR	Adobe AIR
APK (Alpine)	Alpine Linux[2]
APK (Android)	Android
AppImage	Linux distribution-agnostic.
APPX and APPXBundle	Windows 8 and later, Windows Phone[3]
Deb	Debian and its derivatives, such as Ubuntu, Xubuntu, and Linux Mint[4]
ebuild	Gentoo Linux
eopkg	Solus
Flatpak	Linux distribution-agnostic.
HAP and .app	HarmonyOS
PISI	Pardus
PKG	OS X, iOS, PlayStation 3, Solaris, SunOS, UNIX System V, Symbian, BeOS, Apple Newton
.pkg.tar.zst	Arch Linux
PUP and PET	Puppy Linux (PUP format is deprecated since version 3.0)
RPM	Red Hat Enterprise Linux, Fedora, CentOS, SUSE Linux Enterprise Server, openSUSE
Snap	Linux distribution-agnostic, mainly developed for Ubuntu
Windows Installer package / MSI	Windows Installer on Microsoft Windows

Linux Distributives



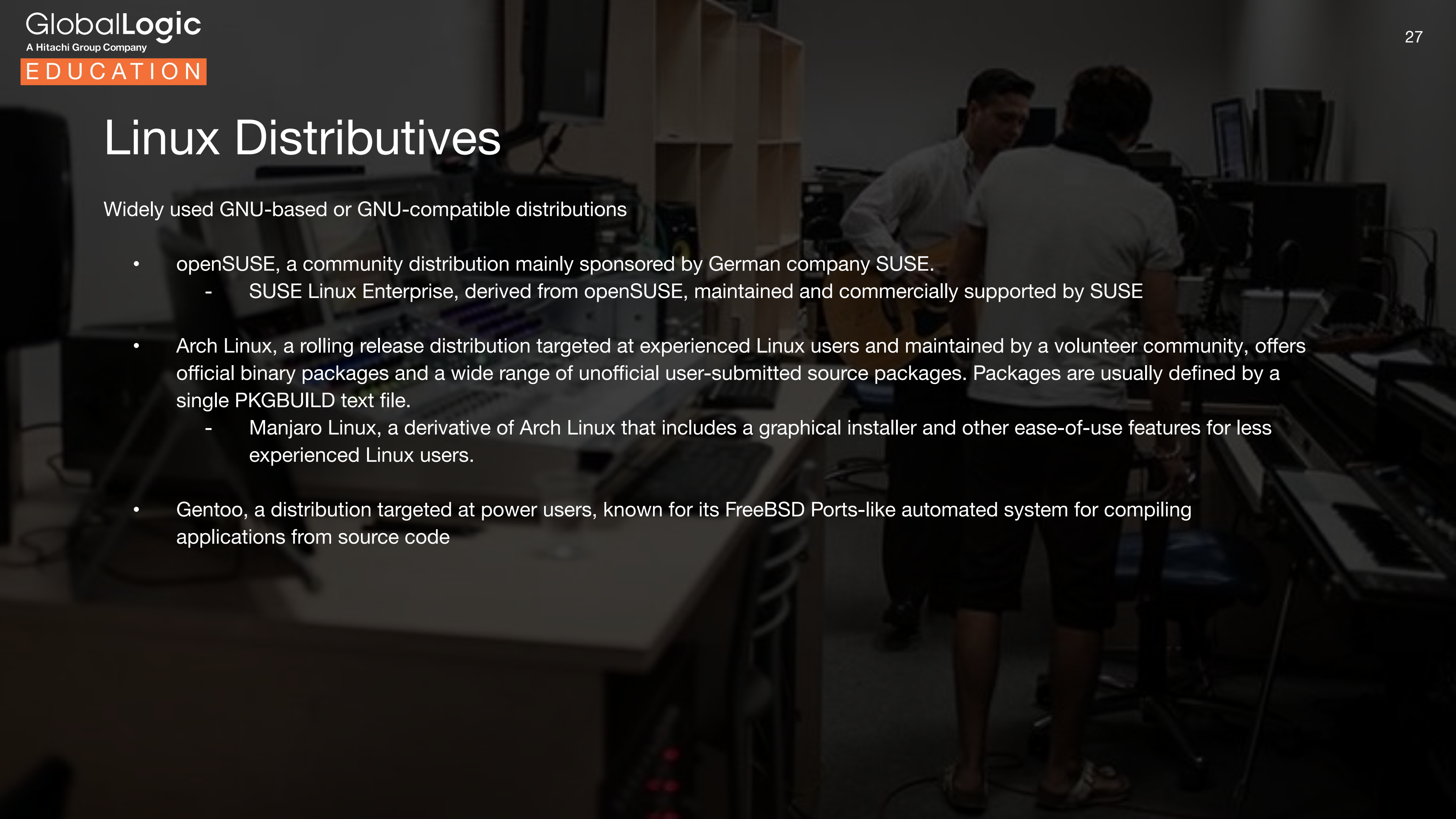


Linux Distributives

Widely used GNU-based or GNU-compatible distributions

- Debian, a non-commercial distribution and one of the earliest, maintained by a volunteer developer community with a strong commitment to free software principles and democratic project management.
  - Ubuntu, a desktop and server distribution derived from Debian, maintained by British company Canonical Ltd.
  - There are several distributions based on Ubuntu that mainly replace the GNOME stock desktop environment, like: Kubuntu based on KDE, Lubuntu based on LXQT, Xubuntu based on XFCE, Ubuntu MATE based on MATE, Ubuntu Budgie based on Budgie. Other official forks have specific uses like: Ubuntu Kylin for Chinese-speaking users, or Ubuntu Studio for media content creators.
  - Linux Mint, a distribution based on and compatible with Ubuntu. Supports multiple desktop environments, among others GNOME Shell fork Cinnamon and GNOME 2 fork MATE.

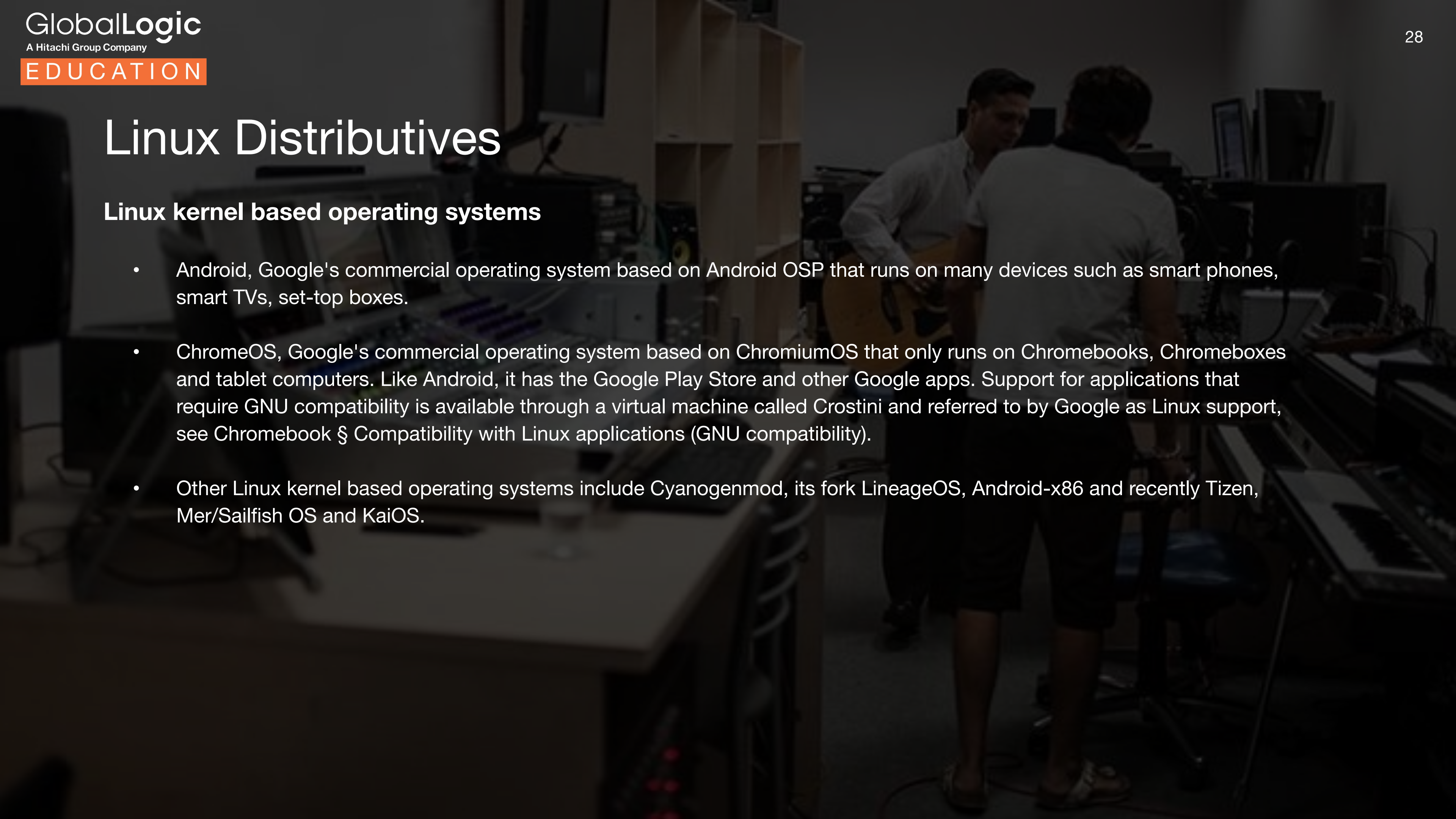
- Fedora Linux, a community distribution sponsored by American company Red Hat and the successor to the company's previous offering, Red Hat Linux. It aims to be a technology testbed for Red Hat's commercial Linux offering, where new open-source software is prototyped, developed, and tested in a communal setting before maturing into Red Hat Enterprise Linux.
  - Red Hat Enterprise Linux (RHEL), a derivative of Fedora Linux, maintained and commercially supported by Red Hat. It seeks to provide tested, secure, and stable Linux server and workstation support to businesses.



Linux Distributives

Widely used GNU-based or GNU-compatible distributions

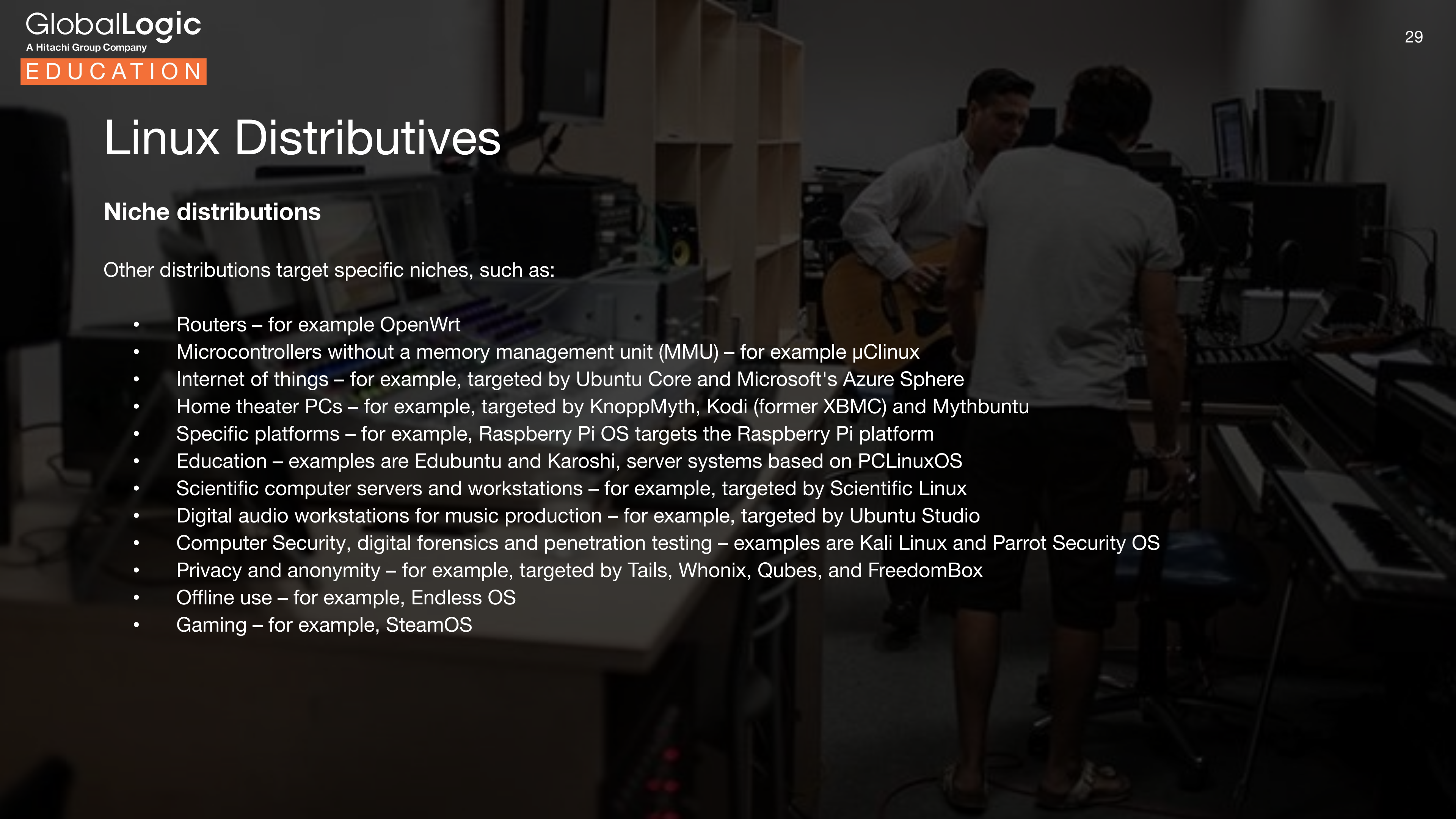
- openSUSE, a community distribution mainly sponsored by German company SUSE.
  - SUSE Linux Enterprise, derived from openSUSE, maintained and commercially supported by SUSE
- Arch Linux, a rolling release distribution targeted at experienced Linux users and maintained by a volunteer community, offers official binary packages and a wide range of unofficial user-submitted source packages. Packages are usually defined by a single PKGBUILD text file.
  - Manjaro Linux, a derivative of Arch Linux that includes a graphical installer and other ease-of-use features for less experienced Linux users.
- Gentoo, a distribution targeted at power users, known for its FreeBSD Ports-like automated system for compiling applications from source code



Linux Distributives

Linux kernel based operating systems

- Android, Google's commercial operating system based on Android OSP that runs on many devices such as smart phones, smart TVs, set-top boxes.
- ChromeOS, Google's commercial operating system based on ChromiumOS that only runs on Chromebooks, Chromeboxes and tablet computers. Like Android, it has the Google Play Store and other Google apps. Support for applications that require GNU compatibility is available through a virtual machine called Crostini and referred to by Google as Linux support, see Chromebook § Compatibility with Linux applications (GNU compatibility).
- Other Linux kernel based operating systems include Cyanogenmod, its fork LineageOS, Android-x86 and recently Tizen, Mer/Sailfish OS and KaiOS.



Linux Distributives

Niche distributions

Other distributions target specific niches, such as:

- Routers – for example OpenWrt
- Microcontrollers without a memory management unit (MMU) – for example μClinux
- Internet of things – for example, targeted by Ubuntu Core and Microsoft's Azure Sphere
- Home theater PCs – for example, targeted by KnoppMyth, Kodi (former XBMC) and Mythbuntu
- Specific platforms – for example, Raspberry Pi OS targets the Raspberry Pi platform
- Education – examples are Edubuntu and Karoshi, server systems based on PCLinuxOS
- Scientific computer servers and workstations – for example, targeted by Scientific Linux
- Digital audio workstations for music production – for example, targeted by Ubuntu Studio
- Computer Security, digital forensics and penetration testing – examples are Kali Linux and Parrot Security OS
- Privacy and anonymity – for example, targeted by Tails, Whonix, Qubes, and FreedomBox
- Offline use – for example, Endless OS
- Gaming – for example, SteamOS

Users, Groups, Passwords

- User Info
  - Current user
    - `$ whoami`
    - `$ id`
    - `$ groups`
    - `$ echo $USER`
    - `$ chsh`
    - `$ finger $USER`
  - Other users
    - root
    - /etc/passwd
    - /etc/group
    - /etc/shadow #type of hash function(md5   etc.) = hash(pass + salt(rand))
    - `$ id` *`username`*
    - `$ finger` *`username`*
    - `$ sudo useradd -m` *`username`* `-s /bin/bash`
    - `$ sudo usermod -a -G cron,crontab` *`username`*
    - `$ sudo userdel` *`username`*

- Logging in
    - Remote
        - PuTTY
            - ~~telnet (obsolete)~~
            - SSH
        - console
            - `$ ssh`
                - `$ ssh` *`myuser@iceboat.cc.telcordia.com`*
            - ~~`$ telnet # obsolete`~~
                - ~~`$ telnet` *`iceboat.cc.telcordia.com`*~~
    - Local
        - console
            - `$ login # root only`
            - `$ su`
                - `$ su -`
                - `$ su` *`username`*
                - `$ su -` *`username`*
                - Ubuntu
                    - root account is locked by default, use the following command instead:
                        - `$ sudo -i/sudo -s`

- password change
  - `$ passwd`
  - `$ passwd` *`username`* `# under root`

```
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
…
user:x:1000:1000:Name 2nd Name,,,:/home/username:/bin/bash
```

```
cat /etc/group
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,user
tty:x:5:
disk:x:6:user
lp:x:7:user
mail:x:8:
news:x:9:
…
user:x:1000:
```

File Attributes

File attributes

- `$ ls -l`
- `$ stat file.txt`
- file type (regular file, directory, symbolic link, socket, fifo)
  - `$ file file.txt`
- modes (access rights, bitmode), rwx:
  - chmod
    - a - all
    - u - user
    - g - group
    - o - other
  - `$ chmod a+x file.txt`
    - `$ chmod +x file.txt`
  - `$ chmod g-w file.txt`
  - `$ chmod u+x file.txt`
  - `$ chmod o-r file.txt`
  - `$ chmod 755 file.txt #r - 4, w - 2, x - 1`
  - `$ chmod 0755 file.txt #r - 4, w - 2, x - 1`

Special bits

- sticky bit (for directories).
  - t equals to 1
  - Set up directory for writing by any user
  - `$ chmod +rwx directory`
  - `$ ls -ld directory`

```
drwxrwxrwx 3 user group 4096 Mar 13 12:34  directory
```

  - **Block** file deletion from this directory by other users:
    - `$ chmod +t directory`
    - `$ ls -ld directory`

```
drwxrwxrwt 3 user group 4096 Mar 13 12:34  directory
```

    - `# user 1 creates file under  directory`
    - `# user 2 tries to remove file from  directory and fails`
  - **Allow** file deletion from this directory by other users:
    - `$ chmod -t directory`
    - `$ ls -ld directory`

```
drwxrwxrwx 3 user group 4096 Mar 13 12:34  directory
```

    - `# user 1 creates file under  directory`
    - `# user 2 tries to remove file from  directory and succeeds`

Special bits

- s-bit (SUID, SGID) bits (for files)
  - Real user ID and group ID, effective user ID and group ID
  - Does not work for executable Shell scripts. Works for binaries.
  - `$ ls -l /usr/bin/passwd`
  - s equals to 4 for user, to 2 for group
  - Set executable bit for binary by file owner (user)

```
● $ whoami
user
● $ chmod +x /home/user/bin/test_sbit
● $ ls -l test_sbit
  -rwxr-xr-x 1 user group 7568 Mar 20 08:26 test_sbit
```

  - **Allow** changing effective user and group IDs for test_sbit utility:

● Set s-bit by file owner (user)

```
● $ whoami
user
● $ chmod +s /home/user/bin/test_sbit
● $ ls -l test_sbit
-rwsr-sr-x 1 user group 7568 Mar 20 08:26 test_sbit
```

Special bits

- s-bit (SUID, SGID) bits (for files)
  - Run the binary by other user (user2)

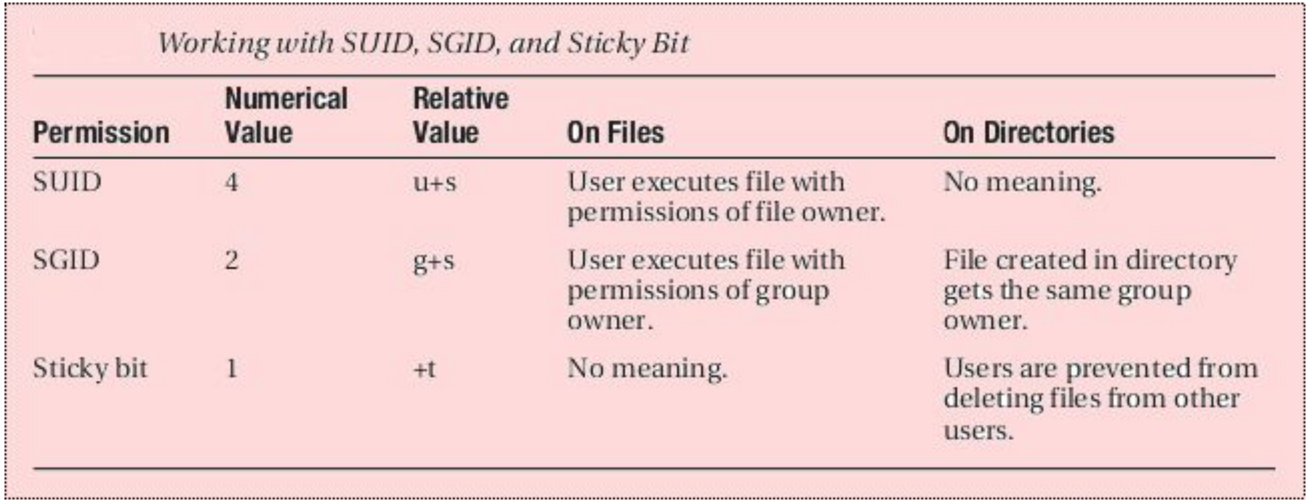
```
$ whoami
user2
$ /home/user/bin/test_sbit
          Real      user & group: user2, group2
Effective user & group:  user, group
```

  - **Disallow** changing effective user and group IDs for test_sbit utility:
  - Remove s-bit by file owner (user)

```
$ whoami
user
$ chmod -s /home/user/bin/test_sbit
$ ls -l test_sbit
-rwxr-xr-x 1 user group 7568 Mar 20 08:26 test_sbit
```

  - Run the binary by other user (user2)

```
$ whoami
user2
$ /home/user/bin/test_sbit
          Real      user & group: user2, group2
Effective user & group:  user2, group2
```



Working with SUID, SGID, and Sticky Bit

Permission	Numerical Value	Relative Value	On Files	On Directories
SUID	4	u+s	User executes file with permissions of file owner.	No meaning.
SGID	2	g+s	User executes file with permissions of group owner.	File created in directory gets the same group owner.
Sticky bit	1	+t	No meaning.	Users are prevented from deleting files from other users.

- owner, group, other users:
  - `$ chown user` *`file.txt`* `# root only`
  - `$ chown :group` *`file.txt`* `# root only`
  - `$ chown user:group` *`file.txt`* `# root only`
  - `$ chown user:group` *`directory`* `# root only`
    - `$ chown -R user:group` *`directory`* `# recursively`
- latest modification time

GlobalLogic
A Hitachi Group Company
EDUCATION

Processes vs Daemons

- Processes vs daemons
  - daemon is a process
  - `/etc/init.d`
  - show running daemons
    - `$ /etc/init.d/`*`daemon_name`*` status`
    - `$ /etc/init.d/ssh status`
  - start daemon automatically
    - `/etc/rc[0-6].d/`
  - run command as a daemon:
    - `$ nohup `*`command`*` &`

- ## Viewing
  - Init daemon OR systemd daemon, PID 1
    - `$ ps -f 1`
      ```
      UID          PID  PPID  C STIME TTY      STAT   TIME CMD
      root           1     0  0 04:41 ?        Ss     0:00 /sbin/init
      ```
    - `$ realpath /sbin/init`
      ```
      /lib/systemd/systemd
      ```
    - `$ sudo realpath /proc/1/exe`
      ```
      /lib/systemd/systemd
      ```
  - `$ ps`
    - `$ ps ux`
    - `$ ps aux`
    - ``$ ps -u `whoami` ``
    - `$ ps -ef`
    - `$ ps axjf`
  - `$ top`
  - `$ htop`

Background Processes

Background processes

- Ctrl-Z
  - `$ sleep 20 # press Enter, then Ctrl-Z`
  - `SIGTSTP/SIGCONT`
- Detaching with &
  - `$ sleep 30 &`
- List of jobs in the current Shell session
  -
    ```
    $ jobs -l
    [1]  + 29797 suspended  sleep 20
    [2]  - 29799 running    sleep 30
    ```
- Run in background
  - `$ bg 1`
- Run in foreground
  - `$ fg 2`
- When exiting from the Shell, SIGHUP may be sent to its attached child processes
  - Check current setting
    - `$ shopt huponexit`
  - Enable SIGHUP
    - `$ shopt -s huponexit`
  - Disable SIGHUP
    - `$ shopt -u huponexit`
  - Ignore SIGHUP by a Shell's child process
    - `$ nohup sleep 60 &`

Signals and Killing

- Signals and killing
  - `$ man 7 signal`
    - `9 : SIGKILL - cannot be caught`
    - `15: SIGTERM - can be caught`
    - ` 1: SIGHUP - can be caught`
    - ` 2: SIGINT - can be caught (Control-C from keyboard)`
  - `$ kill`
    - default signal - SIGTERM (15)
    - `$ kill` *`PID`* `# SIGTERM`
    - `$ kill -9` *`PID`* `# SIGKILL`
    - `$ kill -0` *`PID`* `&& echo "process is alive" || echo "process died"`
    - `$ kill -s` *`signal_name`*
  - `$ killall`
  - Ctrl-C
    - `SIGINT`

Ubuntu Installation to Virtual Machine

- Ubuntu Installation and Configuration under VirtualBox
  - VirtualBox installation and configuration
    - Install VirtualBox (7.0.12) (https://www.virtualbox.org/wiki/Downloads)
    - Configure VirtualBox Host-only network
      - In VirtualBox main window select File -> Host Network Manager
        - In the "Host Network Manager" window add new host-only network adapter. Its name should be set automatically to "vboxnet0" in the LInux OS and “VirtualBox Host-Only Ethernet Adapter” in the Windows OS.
    - Download Ubuntu ISO image
      - http://releases.ubuntu.com/jammy/ubuntu-22.04.3-desktop-amd64.iso
        - Select "amd64" architecture iso link at the very beginning of the page
        - No need to download other images, server for instance.
          - http://releases.ubuntu.com/jammy/ubuntu-22.04.3-live-server-amd64.iso

- Ubuntu Installation and Configuration under VirtualBox
    - VirtualBox installation and configuration
        - Create Virtual Machine in VirtualBox with the following settings:
            - Name: Ubuntu
            - Type: Linux
            - Version: Ubuntu (64-bit)
            - Add hard drive
            - Memory size: 2048 MB
            - Select "Do not add a virtual hard disk" option
        - Configure Ubuntu Virtual Machine:
            - Storage
                - Add optical drive (under IDE controller), choose path on your disk to the Ubuntu ISO downloaded earlier
                - Add SATA drive (under SATA controller) - create a new one
                    - 20GB
                    - VDI
                    - Dynamically allocated
            - Audio
                - disable

- Ubuntu Installation and Configuration under VirtualBox
  - VirtualBox installation and configuration
    - Network
      - For each real network interface in your system enable a corresponding adapter in VirtualBox. Also, configure Ubuntu to use host-only network. Normally, you will need to do the following:
        - Adapter1
          - enable
          - Attached to: Bridged Adapter
          - Name: *name of your **wired** adapter*
        - Adapter2
          - enable
          - Attached to: Bridged Adapter
          - Name: *name of your **wireless** adapter*
        - Adapter3
          - enable
          - Attached to: Host-only Adapter
          - Name: vboxnet0 (Windows OS: VirtualBox Host-Only Ethernet Adapter)

- ## Ubuntu installation
  - Install Ubuntu:
    - Start your empty Ubuntu virtual machine from VirtualBox
    - Installation menu from Ubuntu ISO image should appear with the Four Options: Try or Install Ubuntu, Ubuntu (safe graphics), OEM install, Test memory. Select the 1st one.
      - Try Ubuntu/Install Ubuntu and language selection list.
      - Select a language: English
      - Select "Install Ubuntu"
      - Select Normal Installation option
      - Skip both checkboxes: “Download updates while installing Ubuntu” and “Install third-party software for graphics and Wi-Fi hardware….”
      - Select “Erase Disk and install Ubuntu”
      - Confirm partition changes by pressing “Continue”

- Ubuntu installation
  - Install Ubuntu:
    - Select your location: $Your_Coutry Time
    - Configure Keyboard Layout
      - Keymap to use: English (US)
    - Set up main user:
      - Your name: Tony Stark (example)
      - Your computer’s name: avenger (example)
      - Pick a username: ironman (example)
      - Chose password:
      - Confirm your password:
      - Select “Require my password to log in”
    - Wait until Installation process is completed.
    - Finish installation by restarting the computer, press “Restart now”
    - Please remove the installation medium, then press ENTER:

- Ubuntu runtime configuration
  - Run Ubuntu, log in under non-root user (ironman)
  - Skip Online Accounts Setup
  - Skip Ubuntu Pro Installation Step
  - Select “No, don’t send system info” on “Help improve Ubuntu.
  - Skip Location Services activation
  - Done.
  - Optional: Perform Ubuntu 22.04 Update at appropriate pop-up window if appeared.
  - Run Terminal(terminal will be run with user **ironman**):
    - Log in as root in the terminal:
      - `$ Switch to root with (sudo -s/sudo -i) (use ironman user password)`
      - `Change root user password: $ passwd root`
      - `Change user ID to root $ su -`
      - Enter root's password

- Ubuntu runtime configuration
  - Configure network
    - Check network interfaces
      - `$ ip a`
      - Network interfaces will be printed. Their names should look like: enp0s3, enp0s8, enp0s9.
        - enp0s3 and enp0s8 should be linked with host machine's network adapters (wired and wireless)
        - enp0s9 should be linked with VirtualBox's host-only adapter.
        - NOTE: ethX interfaces names are not used anymore since Predictable Network Interface naming is used starting from version 15.04 and is part of systemd(version 197), to which Ubuntu has been transitioning:

```
enp4s10f1                      pci 0000:04:0a.1
| | |  |                                | | | |
| | |  |                 domain <- 0000   | | |
| | |  |                                  | | |
en| |  | --> ethernet                     | | |
           | |  |                                  | | |
           p4|  | --> prefix/bus number (4)  <-- 04 | |
                    |  |                                    | |
                    s10| --> slot/device number (10) <-- 10   |
                 |                                      |
                 f1 --> function number (1)   <--       1
```

Ubuntu runtime configuration

- Configure network
  - Verify that you are connected to the Internet from virtual machine:
    - `$ ping -c 4 8.8.8.8`
  - Check network interfaces linked
    - enp0s3 - wired interface
      - Make sure the interface is up and IP address is set
        - `$ ip a show dev` enp0s3
        - Verify that IP address is **not** from `169.254.*.*` range.
      - Make sure the Internet is available via this interface
        - `$ ping -I` enp0s3 `-c 4 8.8.8.8`
    - enp0s8 - wireless interface
      - Make sure the interface is up and IP address is set
        - `$ ip a show dev` enp0s8
        - Verify that IP address is **not** from `169.254.*.*` range.
      - Make sure the Internet is available via this interface
        - `$ ping -I` enp0s8 `-c 4 8.8.8.8`

- Ubuntu runtime configuration
  - Configure network
    - Check network interfaces linked
      - enp0s9 - host-only interface
        - Make sure the interface is up and IP address is set
          - `$ ip a show dev` enp0s9
          - Verify that IP address is `192.168.56.X`
        - Make sure the Internet is **not** available via this interface
          - `$ ping -I` enp0s9 `-c 4 8.8.8.8`
        - Verify that you are connected to the host machine from the virtual one:
          - `$ ping -I` enp0s9`-c 4 192.168.56.1`
  - Install ifconfig utility in Ubuntu 22.04 ($ sudo apt-get install net-tools)
    - net-tools is needed in order to install **ifconfig**.
    - **ifconfig** is a command used to configure network interfaces on Linux machines. However, it is now considered a deprecated command because it has been replaced by **ip** command, which offers more advanced features and better control over network configuration.

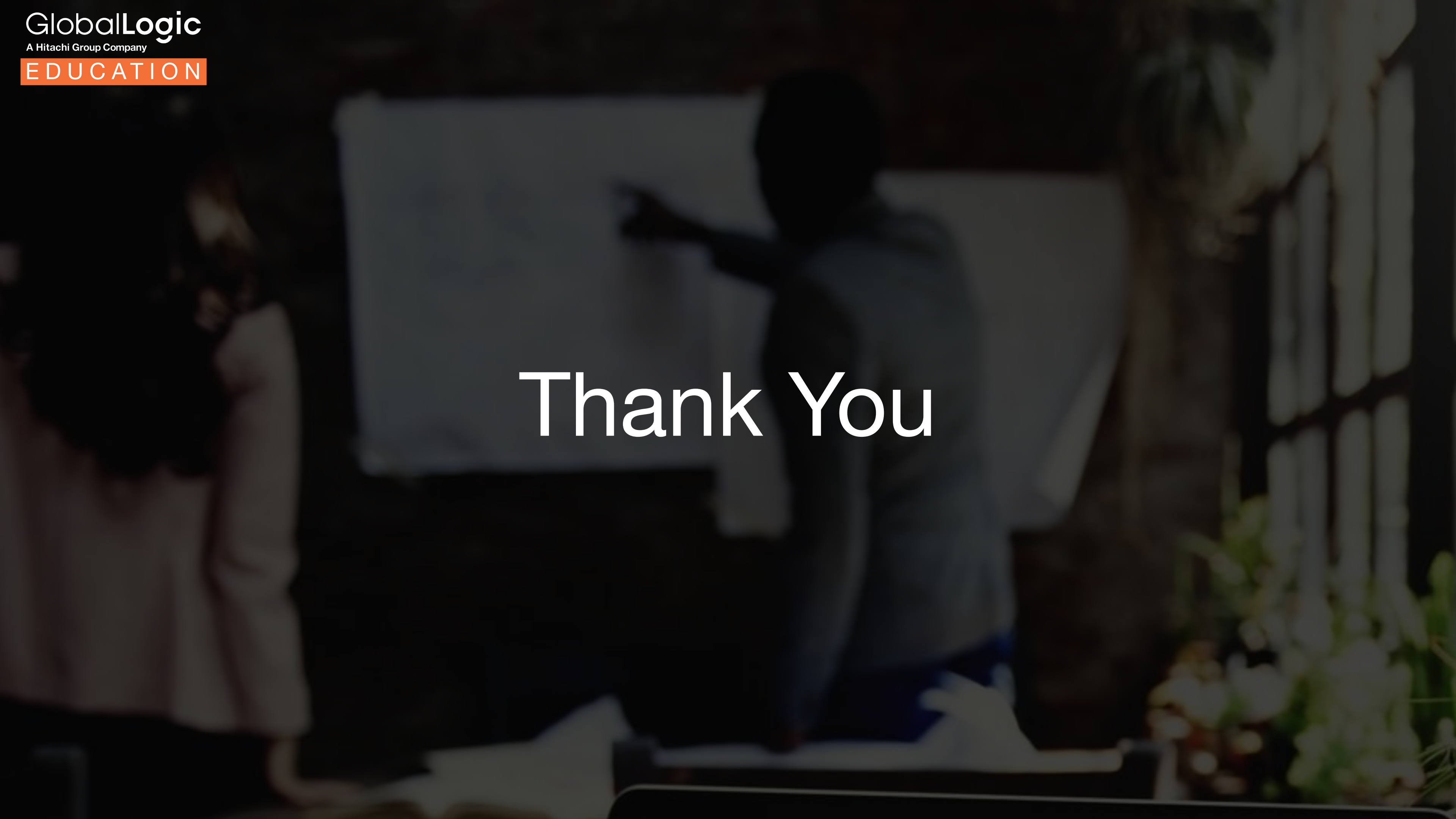
- Ubuntu runtime configuration
  - Configure network
    - Install SSHD service in Ubuntu virtual machine:
      - `$ sudo apt-get install openssh-server`
    - `Verify whether SSHD service is up and running after installation:`
      - `$ /etc/init.d/ssh status`
      - If not started, start:
        - `$ sudo /etc/init.d/ssh start`
- Remote connection to Ubuntu
  - Verify you can login via SSH **from host machine to virtual Ubuntu machine**
    - PuTTY
      - SSH
        - host: `192.168.56.104`
        - user: `ironman`
    - console
      - `$ ssh ironman@192.168.56.104`

GlobalLogic
A Hitachi Group Company
EDUCATION

Q&A

1. Question:

GlobalLogic
A Hitachi Group Company
EDUCATION



Thank You