Operating Systems – User interfaces and demos

Péter Györke
http://www.mit.bme.hu/~gyorke/
gyorke@mit.bme.hu

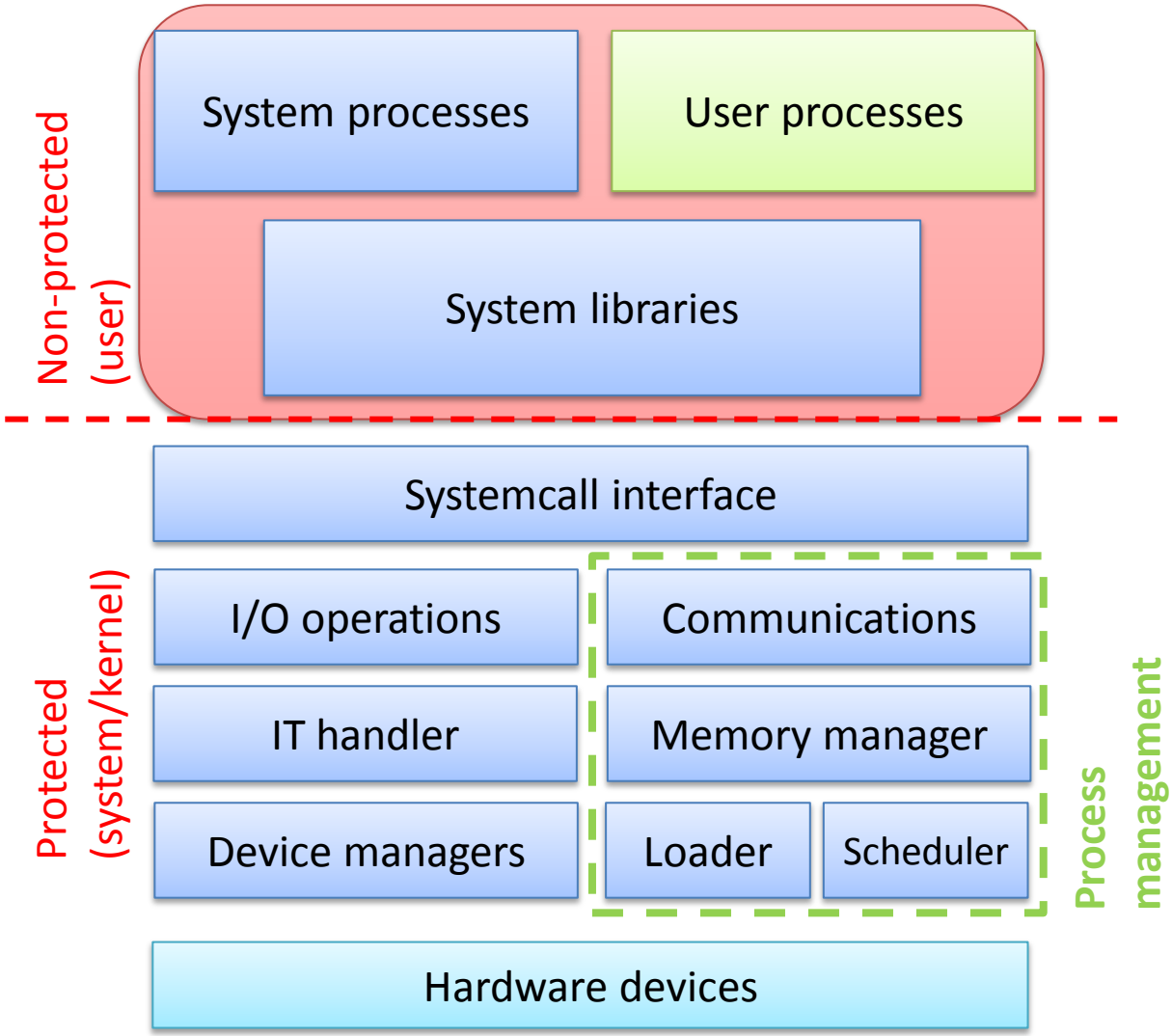
Budapest University of Technology and Economics (BME)
Department of Measurement and Information Systems (MIT)

The slides of the latest lecture will be on the course page. (https://www.mit.bme.hu/eng/oktatas/targyak/vimiab00)

The operating systems (recap)

- Serving user (and system) tasks
  - Life-cycle (creation, operation, termination) and event monitoring
  - Providing computational and storage resources
  - Providing access to the devices of the computer
- Types (incomplete list)
  - Client OS: usually with GUI
  - Server OS: usually with console only
- System applications and services
  - User mode programs which comes with the OS
  - Integrated **commands** and **user interfaces**, **services**

The main blocks of the OS and the kernel



Requirements of a user interface

- Processing the commands of the user
  - Preprocessing the command (find the executable program)
    - Built-in commands: belongs to the user interface
    - External commands: external programs for executing the user tasks
      - These can be also part of the OS: system application
      - Third party software installed on the system
      - Maybe a software developed by the user
- Starting the program to execute the command
  - The user interface creates a new process and starts the program
  - It may pass arguments to the process (e.g. `argv[]`)
- Connecting the user with the task
  - When the task is running it is connected with the user
  - The user interface provides an environment to the tasks connected with the user
  - The user interface manages the user's session (a set of tasks)
  - It returns the results of the tasks, or the errors if there's any
  - It shows the user interface of the task, if it has one
- User friendly behaviour

Session

- Set of tasks connected to an activity
  - 0. session: set of OS services (see booting)
  - 1+ sessions: sessions of the active users
    - Usually one session per one user login
- The session
  - The goal is to group the tasks
    - Example: if the user logs out, the user's tasks can be closed
  - The session manager handles the process set
  - Usually there is a user interface (terminal) for the session manager
  - Some user interfaces are able to save and restore the previous state
  - It may has multiple process groups (e.g.: foreground and background processes)
- Process group
  - For processes which are belongs together
  - It has a manager (controller), which initiates the processes
  - The whole group is **notified about certain events**
  - The user can control the whole group as one unit

User interface types: Graphical user interface

- Graphical user interface (GUI, windowing system)
  - It shows visual elements (icons, menus, etc.) created by pixels
  - The command interpreter has GUI
  - It can be controlled via keyboard, mouse or touchscreen
  - The user interfaces of the applications are showed in windows (windowing system)
  - The windows are managed by the **window manager**
  - The window manager is served by the display server
  - Total user experience
  - Not available in every system (economy, complexity reduction)
- Others
  - Voice activated
  - Controlled by natural language

GUI and windowing system

- Graphical user interface (GUI)
  - Graphical version of the shell
  - Complex, typically layered architecture with open interfaces
  - Typical blocks: command interpreter, display server, window manager
  - Examples: Windows shell, Gnome Shell, Ubuntu Unity, KDE, LXDE, etc.
- Window manager (WM)
  - Controls the appearance and placement of the application windows
  - The user is able to change windows
  - The visual appearance is customizable by the user
  - Examples
    - Windows Vista and later: Desktop Window Manager (dwm.exe)
    - Linux: KDE-Kwin, Gnome2-Metacity, etc.

User interface types: Character terminal

- Character terminal
  - The user is connected to a **shell** (command interpreter)
  - The commands typically has an stdin input and stdout output, or may have other interfaces
  - It is available in every system
  - The user can connect via: keyboard+monitor, from network, from serial port, etc.
  - The user experience is rather limited (not every application is able to run in a char. terminal)
  - It is an efficient interface for administration (it can be programmed)

Character terminals

- Interpreting and executing commands
  - UNIX: bash, csh, ksh, zsh, etc.
  - Windows: cmd.exe (later powershell)
- Built-in commands
  - Job control
  - Simple text output and input
  - Bash built-ins: `logout, alias, echo, read, source, ulimit,` etc.
  - Powershell keywords, Cmdlets, core and external modules (for many applications)
- It can be programmed!
  - It is an efficient tool to manage the OS
  - Powerful text managing (e.g.: log processing)
  - The terminal's interpreter provides standard programming language elements: if-else, loops, functions, macros, etc.
  - The „programs" may use external commands: almost every installed application which has a command line interface
- Built-in help

UNIX

```
man <command>
<command> -h or--help
```

Windows

```
Get-Help <command>
<command> /h  or /?
```

„What happens in the system?”

- Listing running tasks (UNIX)
  - `ps, ps -ef, ps axu, ps -u <user>, pstree, ...`
  - `top, atop, htop` and others
- Administrative parameters of the tasks
  - Unique PID (Process ID), PPID: PID of the parent
  - The state of the process (running, waiting, etc.)
  - Scheduling information (e.g.: priority, see later)
  - Authenticators
  - Memory management data
  - Communication data
  - TTY: which terminal (user login) is connected to the process
  - STIME: when did the task started
  - TIME: running time on the CPU
  - CMD: the command (and arguments) which started the process
  - …

Programming the shell

- Interpreter (script) languages (no compile)
- Long evolution
  - The functions are extended by the user demands
  - In some cases they are overcomplicated
- Language elements of the shell
  - Built-in elements:
    - Programming structures (if-else, loops, etc.)
    - Simple OS tasks (list files, task management, etc.)
  - External commands
    - Every installed application with a command line interface
    - Many text processing tools (to process the outputs of the commands)
      - grep, sed, awk, sort, uniq
  - Extension modules
    - Windows powershell cmdlets
    - It extends the abilities of the shell as a module
- Combining commands
  - The commands can be **connected via pipes**
  - Example:

```
du -s * | sort -n
Get-Service | Where-Object {$_.DependentServices -ne $null}
```

Shell script examples

- The command can be written in a file
- Variables

```
#!/bin/bash
# variable declaration
TEXT="scripts are fun"
# writing the variable to the output
echo $TEXT
```

- Getting IP address

  `ifconfig` – for getting network interface status

  `ifconfig | awk '/inet addr/{print substr($2,6)},`

- `(awk '/search_pattern/ { action_to_take_on_matches; another_action; }' file_to_parse)`
- `Making it better: ifconfig | grep -A 1 "eth0" | awk '/inet addr/{print substr($2,6)},`

- Sending the IP address to a (web) server
  - For some kind of dynamic DNS service
  - `curl 91.82.85.156/ping/put.php?ip=12.12.12.12`
- Scheduling this script to run periodically
  - CRON

How to try these at home?

- VirtualBox: www.virtualbox.org
  - Download prepared boxes: www.virtualboxes.org
    - Only for experimenting, for critical application a self installed system should be set up, which is downloaded from a trusted source (with MD5 or SHA check)
  - Install new blank machine from ISO image

Inspecting kernel data structures (see task management)

- The kernel data structures can be accessed through the file system (read-only)
  - The field `ctxt` of the file `/proc/stat` shows the number of context changes
  - It can be listed for a specific process: `/proc/<PID>/status`
    - `voluntary_ctxt_switches` and `nonvoluntary_ctxt_switches` fields

- Performing Apache2 (webserver) load benchmark
  - Observe the number of context changes of the `apache2` process
  - What is the nature of the process `apache2`?

- Observing the context changes of a CPU intensive process

  For example: `stress -c 1`
  - Check the context changes of the child process of stress
  - How does the field `nonvoluntary_ctxt_switches` change?
  - According to this: what is the scheduler type of the OS?

- These experiments can be performed under Windows also

Commands for Apache Benchmarking

- `apache2 -V | grep -i 'version\|mpm'`
- `/etc/apache2/mods-available/`
- `sudo a2dismod mpm_event`
- `sudo a2enmod mpm_prefork`
- `sudo systemctl restart apache2`
- `ab -n 9000 -c 300 http://localhost/`