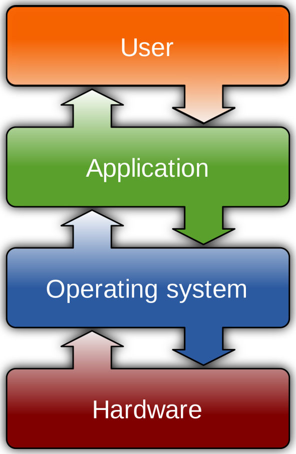
OS security - OS key concepts

Myrto Arapinis
School of Informatics
University of Edinburgh

March 6, 2019

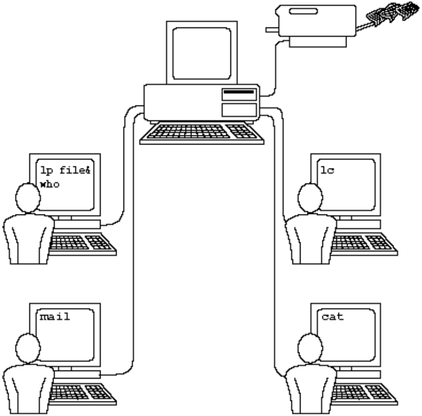
Operating systems

- An OS provides the interface between the users of a computer and that computer's hardware.
- The OS handles the management of low-level hardware resources:
  - disk drives,
  - CPU,
  - RAM,
  - I/O devices, and



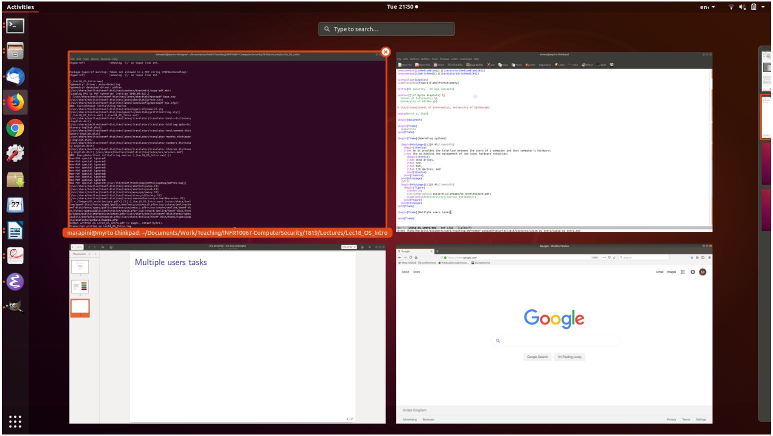
Multi-users

OSs must allow for multiple users with potentially different levels of access to the same computer.



Multi-tasking

OSs must allow multiple application programs to run at the same time.



Activities
Tue 21:50
en
Type to search...
27
marapini@myrto-thinkpad: ~/Documents/Work/Teaching/INFR10067-ComputerSecurity/1819/Lectures/Lec18_OS_Intro
File Edit View Search Terminal Help
emacs@myrto-thinkpad
File Edit Options Buffers Tools Preview LaTeX Command Help
New File Open File Dired Kill Buffer Save Buffer Cut Copy Paste Undo Separator Latex View
OS security - OS key concepts
Thumbnails
Google - Mozilla Firefox
Google
https://www.google.com
Most Visited Conferences Publication submissio... CS166/CS162
About Store
Gmail Images
Google
Google Search I'm Feeling Lucky
United Kingdom
Advertising Business
Privacy Terms Settings

```
(hyperref)                removing `\\' on input line 147.


Package hyperref Warning: Token not allowed in a PDF string (PDFDocEncoding):
(hyperref)                removing `\\' on input line 147.

(./Lec18_OS_Intro.aux)
*geometry* driver: auto-detecting
*geometry* detected driver: pdftex
(/usr/share/texlive/texmf-dist/tex/context/base/mkii/supp-pdf.mkii
[Loading MPS to PDF converter (version 2006.09.02).]
) (/usr/share/texlive/texmf-dist/tex/latex/oberdiek/epstopdf-base.sty
(/usr/share/texlive/texmf-dist/tex/latex/oberdiek/grfext.sty)
(/usr/share/texlive/texmf-dist/tex/latex/latexconfig/epstopdf-sys.cfg))
ABD: EveryShipout initializing macros
(/usr/share/texlive/texmf-dist/tex/latex/hyperref/nameref.sty
(/usr/share/texlive/texmf-dist/tex/generic/oberdiek/gettitlestring.sty))
(./Lec18_OS_Intro.out) (./Lec18_OS_Intro.out)
(/usr/share/texlive/texmf-dist/tex/latex/translator/translator-basic-dictionary
-English.dict)
(/usr/share/texlive/texmf-dist/tex/latex/translator/translator-bibliography-dic
tionary-English.dict)
(/usr/share/texlive/texmf-dist/tex/latex/translator/translator-environment-dict
ionary-English.dict)
(/usr/share/texlive/texmf-dist/tex/latex/translator/translator-months-dictionar
y-English.dict)
(/usr/share/texlive/texmf-dist/tex/latex/translator/translator-numbers-dictiona
ry-English.dict)
(/usr/share/texlive/texmf-dist/tex/latex/translator/translator-theorem-dictiona
ry-English.dict) (/usr/share/texlive/texmf-dist/tex/latex/ucs/ucsencs.def)
ABD: EverySelectfont initializing macros (./Lec18_OS_Intro.nav) [1
Non-PDF special ignored!
Non-PDF special ignored!
Non-PDF special ignored!
Non-PDF special ignored!
Non-PDF special ignored!
Non-PDF special ignored!
Non-PDF special ignored!
Non-PDF special ignored!
Non-PDF special ignored!{/var/lib/texmf/fonts/map/pdftex/updmap/pdftex.map}]
(/usr/share/texlive/texmf-dist/tex/latex/amsfonts/umsa.fd)
(/usr/share/texlive/texmf-dist/tex/latex/amsfonts/umsb.fd)
(/usr/share/texlive/texmf-dist/tex/latex/wasysym/uwasy.fd)
(/usr/share/texlive/texmf-dist/tex/latex/stmaryrd/Ustmry.fd)
(/usr/share/texlive/texmf-dist/tex/latex/sansmathaccent/ot1mathkerncmss.fd)
[2 <./Images/OS_architecture.pdf>] [3] (./Lec18_OS_Intro.aux) )</usr/share/texl
ive/texmf-dist/fonts/type1/public/amsfonts/cm/cmss10.pfb></usr/share/texlive/te
xmf-dist/fonts/type1/public/amsfonts/cm/cmss12.pfb></usr/share/texlive/texmf-di
st/fonts/type1/public/amsfonts/cm/cmss8.pfb></usr/share/texlive/texmf-dist/font
s/type1/public/amsfonts/cm/cmss9.pfb></usr/share/texlive/texmf-dist/fonts/type1
/public/amsfonts/cm/cmssbx10.pfb></usr/share/texlive/texmf-dist/fonts/type1/pub
lic/amsfonts/symbols/msam10.pfb>
Output written on Lec18_OS_Intro.pdf (3 pages, 249447 bytes).
Transcript written on Lec18_OS_Intro.log.
```

```
\newcommand{\InMediumBlue}[1]{\textcolor{MediumBlue}{#1}}
\newcommand{\InBrickRed}[1]{\textcolor{BrickRed}{#1}}

\usepackage{caption}
\captionsetup[figure]{labelformat=empty}

\title{OS security - OS key concepts}

\author[{\bf Myrto Arapinis} \\
  School of Informatics \\
  University of Edinburgh}

% \institute{School of Informatics, University of Edinburgh}

\date{March 6, 2019}

\begin{document}

\begin{frame}
  \maketitle
\end{frame}

\begin{frame}{Operating systems}

  \begin{minipage}[c]{0.45\linewidth}
    \begin{itemize}
    \item An OS provides the interface between the users of a computer and that computer's hardware.
    \item The OS handles the management of low-level hardware resources:
      \begin{itemize}
      \item disk drives,
      \item CPU,
      \item RAM,
      \item I/O devices, and
      \end{itemize}
    \end{itemize}
  \end{minipage}
  \hfill
  \begin{minipage}[c]{0.45\linewidth}
    \begin{figure}
      \centering
      \includegraphics[scale=0.1]{Images/OS_architecture.pdf}
      \caption{\textcolor{Grey}{source: Wikipedia}}
    \end{figure}
  \end{minipage}
\end{frame}

\begin{frame}{Multiple users tasks}

\end{frame}
U:---  Lec18_OS_Intro.tex    Bot L183    (LaTeX/P)
Wrote /home/marapini/Documents/Work/Teaching/INFR10067-ComputerSecurity/1819/Lectures/Lec18_OS_Intro/Lec18_OS_Intro.tex
```

Multiple users tasks

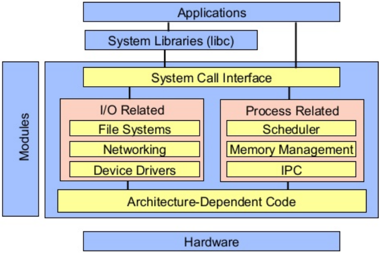
3 / 3

Essential Unix architecture

Execution modes:

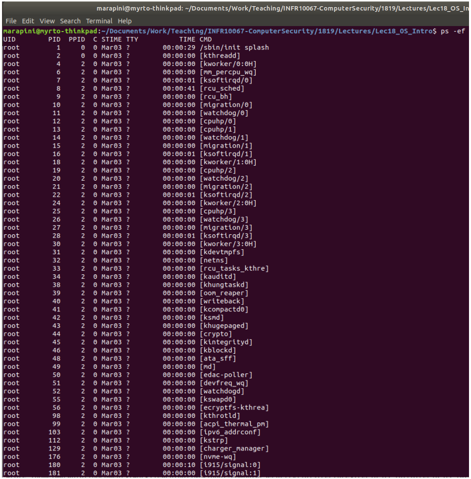
- User mode - access to resources through syscall to kernel
- Kernel mode - direct access to resources

System calls are usually contained in a collection of programs, eg. a library such as the C library `libc`



Processes and process management

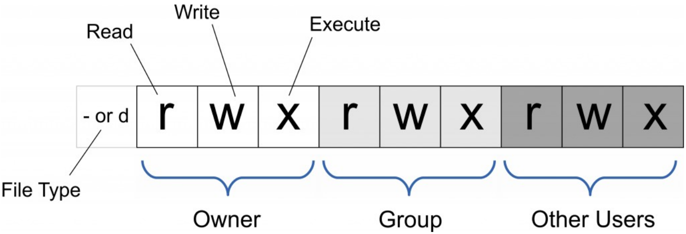
- A process is an instance of a program that is currently executing.
- To actually be executed the program must be loaded into RAM and uniquely identified.
- Each process running is identified by a unique Process ID (PID).
- To a PID, we can associate its CPU time, memory usage, user ID (UID), program name, etc.
- A process might control other processes (fork).
- Child process inherits context from parent process.


marapini@myrto-thinkpad: ~/Documents/Work/Teaching/INFR10067-ComputerSecurity/1819/Lectures/Lec18_OS_In
File
Edit
View
Search
Terminal
Help

```
marapini@myrto-thinkpad:~/Documents/Work/Teaching/INFR10067-ComputerSecurity/1819/Lectures/Lec18_OS_Intro$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Mar03 ?        00:00:29 /sbin/init splash
root         2     0  0 Mar03 ?        00:00:00 [kthreadd]
root         4     2  0 Mar03 ?        00:00:00 [kworker/0:0H]
root         6     2  0 Mar03 ?        00:00:00 [mm_percpu_wq]
root         7     2  0 Mar03 ?        00:00:01 [ksoftirqd/0]
root         8     2  0 Mar03 ?        00:00:41 [rcu_sched]
root         9     2  0 Mar03 ?        00:00:00 [rcu_bh]
root        10     2  0 Mar03 ?        00:00:00 [migration/0]
root        11     2  0 Mar03 ?        00:00:00 [watchdog/0]
root        12     2  0 Mar03 ?        00:00:00 [cpuhp/0]
root        13     2  0 Mar03 ?        00:00:00 [cpuhp/1]
root        14     2  0 Mar03 ?        00:00:00 [watchdog/1]
root        15     2  0 Mar03 ?        00:00:00 [migration/1]
root        16     2  0 Mar03 ?        00:00:01 [ksoftirqd/1]
root        18     2  0 Mar03 ?        00:00:00 [kworker/1:0H]
root        19     2  0 Mar03 ?        00:00:00 [cpuhp/2]
root        20     2  0 Mar03 ?        00:00:00 [watchdog/2]
root        21     2  0 Mar03 ?        00:00:00 [migration/2]
root        22     2  0 Mar03 ?        00:00:01 [ksoftirqd/2]
root        24     2  0 Mar03 ?        00:00:00 [kworker/2:0H]
root        25     2  0 Mar03 ?        00:00:00 [cpuhp/3]
root        26     2  0 Mar03 ?        00:00:00 [watchdog/3]
root        27     2  0 Mar03 ?        00:00:00 [migration/3]
root        28     2  0 Mar03 ?        00:00:01 [ksoftirqd/3]
root        30     2  0 Mar03 ?        00:00:00 [kworker/3:0H]
root        31     2  0 Mar03 ?        00:00:00 [kdevtmpfs]
root        32     2  0 Mar03 ?        00:00:00 [netns]
root        33     2  0 Mar03 ?        00:00:00 [rcu_tasks_kthre]
root        34     2  0 Mar03 ?        00:00:00 [kauditd]
root        38     2  0 Mar03 ?        00:00:00 [khungtaskd]
root        39     2  0 Mar03 ?        00:00:00 [oom_reaper]
root        40     2  0 Mar03 ?        00:00:00 [writeback]
root        41     2  0 Mar03 ?        00:00:00 [kcompactd0]
root        42     2  0 Mar03 ?        00:00:00 [ksmd]
root        43     2  0 Mar03 ?        00:00:00 [khugepaged]
root        44     2  0 Mar03 ?        00:00:00 [crypto]
root        45     2  0 Mar03 ?        00:00:00 [kintegrityd]
root        46     2  0 Mar03 ?        00:00:00 [kblockd]
root        48     2  0 Mar03 ?        00:00:00 [ata_sff]
root        49     2  0 Mar03 ?        00:00:00 [md]
root        50     2  0 Mar03 ?        00:00:00 [edac-poller]
root        51     2  0 Mar03 ?        00:00:00 [devfreq_wq]
root        52     2  0 Mar03 ?        00:00:00 [watchdogd]
root        55     2  0 Mar03 ?        00:00:00 [kswapd0]
root        56     2  0 Mar03 ?        00:00:00 [ecryptfs-kthrea]
root        98     2  0 Mar03 ?        00:00:00 [kthrotld]
root        99     2  0 Mar03 ?        00:00:00 [acpi_thermal_pm]
root       103     2  0 Mar03 ?        00:00:00 [ipv6_addrconf]
root       112     2  0 Mar03 ?        00:00:00 [kstrp]
root       129     2  0 Mar03 ?        00:00:00 [charger_manager]
root       176     2  0 Mar03 ?        00:00:00 [nvme-wq]
root       180     2  0 Mar03 ?        00:00:10 [i915/signal:0]
root       181     2  0 Mar03 ?        00:00:00 [i915/signal:1]
```

File permissions

- One of the main concern of OSs is how to delineate which user can access which resources.
- File permissions are checked by the OS to determine if a file is readable, writable, or executable by a user or a group of users.

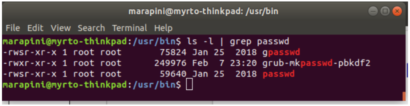


Unix file permissions

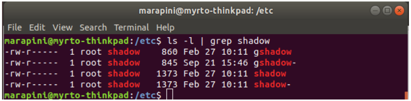
Setuid programs

- Unix process have 2 user IDs:
  - real user ID (uid) - user launching the pgm
  - effective user ID (euid) - user owning the pgm
- An executable file can have the set-user-ID property (setuid) enabled
- If A executes setuid file owned by B, then the euid of the process is B and not A
- Syscall setuid(uid) allows a process to change its euid to uid
- Writing secure setuid programs is tricky because vulnerabilities may be exploited by malicious user actions

- Some programs that access system resources are owned by root and have the setuid bit set (setuid programs)


marapini@myrto-thinkpad: /usr/bin
File Edit View Search Terminal Help

```
marapini@myrto-thinkpad:/usr/bin$ ls -l | grep passwd
-rwsr-xr-x 1 root root       75824 Jan 25  2018 gpasswd
-rwxr-xr-x 1 root root      249976 Feb  7 23:20 grub-mkpasswd-pbkdf2
-rwsr-xr-x 1 root root       59640 Jan 25  2018 passwd
marapini@myrto-thinkpad:/usr/bin$ 
```

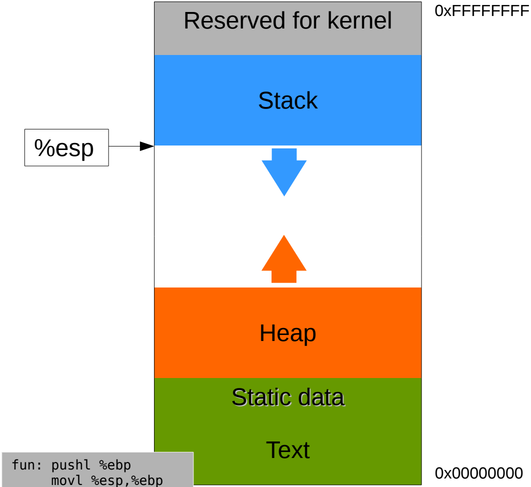

marapini@myrto-thinkpad: /etc
File Edit View Search Terminal Help

```
marapini@myrto-thinkpad:/etc$ ls -l | grep shadow
-rw-r-----  1 root shadow    860 Feb 27 10:11 gshadow
-rw-r-----  1 root shadow    845 Sep 21 15:46 gshadow-
-rw-r-----  1 root shadow   1373 Feb 27 10:11 shadow
-rw-r-----  1 root shadow   1373 Feb 27 10:11 shadow-
marapini@myrto-thinkpad:/etc$ 
```

Memory management

- To actually be executed the program must be loaded into RAM and uniquely identified.
- The RAM memory of a computer is its address space.
- It contains both the code for the running program, its input data, and its working memory.
- For any running process it is organised into different segments, which keep the different parts of the address space separate
- Security concerns require that we do not mix up these different segments.

Linux (32-bit) process memory layout (simplified)

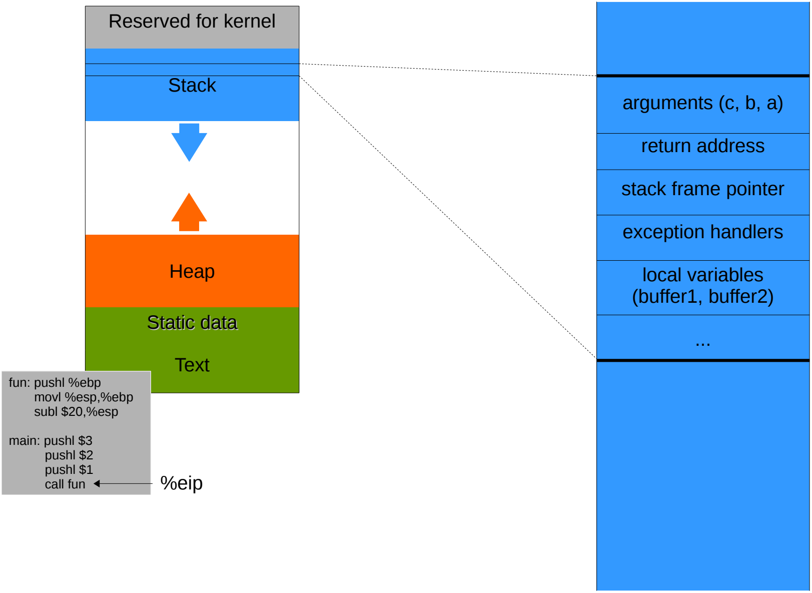


```
void function(int a, int b, int c) {
     char buffer1[5];
     char buffer2[10];
}
void main() {
     function(1,2,3);
}
```

```
fun: pushl %ebp
     movl %esp,%ebp
     subl $20,%esp

main: pushl $3
      pushl $2
      pushl $1
      call fun
```

Stack frame



```
fun: pushl %ebp
       movl %esp,%ebp
       subl $20,%esp

main: pushl $3
          pushl $2
          pushl $1
          call fun
```

Stack and functions: Summary

Calling function

1. Push arguments onto the stack (in reverse)
2. Push the return address, i.e., the address of the instruction to run after control returns
3. Jump to the function's address

Called function

4. Push the old frame pointer onto the stack (%ebp)
5. Set frame pointer (%ebp) to where the end of the stack is right now (%esp)
6. Push local variables onto the stack

Returning function

7. Reset the previous stack frame: %esp = %ebp, %ebp = (%ebp)
8. Jump back to return address: %eip = 4(%esp)