OS Theory Concept Map

Tyler Chotikamars

CPT304: Operating Systems Theory & Design

Pete Limon

3/01/2021

The major functions of an operating system include security, system programs, command interpreters, memory management, process management and secondary-storage management. With this being true it is easy to see how the operating system is made up of a bunch of individual systems that communicate with each other to achieve a single task. Operating systems allow processes to share and exchange information. A process can be defined as, "A process is a program in execution. It also includes the current activity and generally also includes the process stack" (Silberschatz, et. Al., 2014). A process state describes the state of which the process is, they can be, new, ready, running, waiting, halted and so on. Finally, a process control block represents each process in the operating system, this can also be called the task control block. This block shows information that is specific within the process. Multi-threaded systems can multi-task, thus performing more than one task at a single time. On the other hand, single thread processes can only handle one process at a time. A critical-section problem is a segment of code in which the process may be changing common variables, updating a table, writing a file, and so on. "The critical-section problem is to design a protocol that the processes can use to cooperate", (Silberschatz, et. Al., 2014). A software solution that can be used but is not universal is called Peterson's Solution. Peterson's Solution involves two processes that share two data items but alternate execution between their critical sections and remainder sections (Silberschatz, et. Al., 2014).

The basic things that are needed to manage memory include basic hardware, the binding of symbolic memory addresses to actual physical addresses, and the distinction between logical and physical addresses. Within a regular single-tasking procedure one process is selected from the input queue and loads that process into memory. As it is being executed it accesses

instructions and data from the memory. Then it terminates and the memory space is declared available. Logical addresses generated by the CPU are different from memory-address registers in the sense that only the memory unit can see the logical address. The compile-time and load-time address-binding methods generate the same logical and physical address. The set of logical addresses generated is called the logical address space, while the set of all physical addresses corresponding to these logical addresses are known as a physical address space. Leaving the user with an execution-time address-binding scheme and the logical and physical address spaces. While the user probably does not get to see all these addresses, they relate to different memory mapping techniques in operating systems. Logical memory usually gives directions for where memory is being stored within the virtual storage of that memory. Physical memory refers to the hardware address of the memory. "Virtual memory is a technique that enables us to map a large logical address space onto a smaller physical memory", (Silberschatz, et. Al., 2014). The file systems management consists of two parts which are a collection of files and a directory structure. Since a file is an abstract piece of data the file systems management must be able to create, read and write files, as well as deleting, truncating, and repositioning within a file. Simply put, the objectives of a file management system are to manage and store files. Single-level directories are simple and easy to understand but can run into problems when they must store many files or have multiple users. This is because each file must have a unique name within the directory. A two-level directory will create separate directories for each user which separates all the files within their respective directory. A tree-structured directory allows each user to have a subdirectory to allow the directory to branch off. In this format, the tree has a root directory and every file within the system has a unique pathname. Acyclic-graph directories allow users to share a subdirectory to access the same files as well as having their own directories for each user.

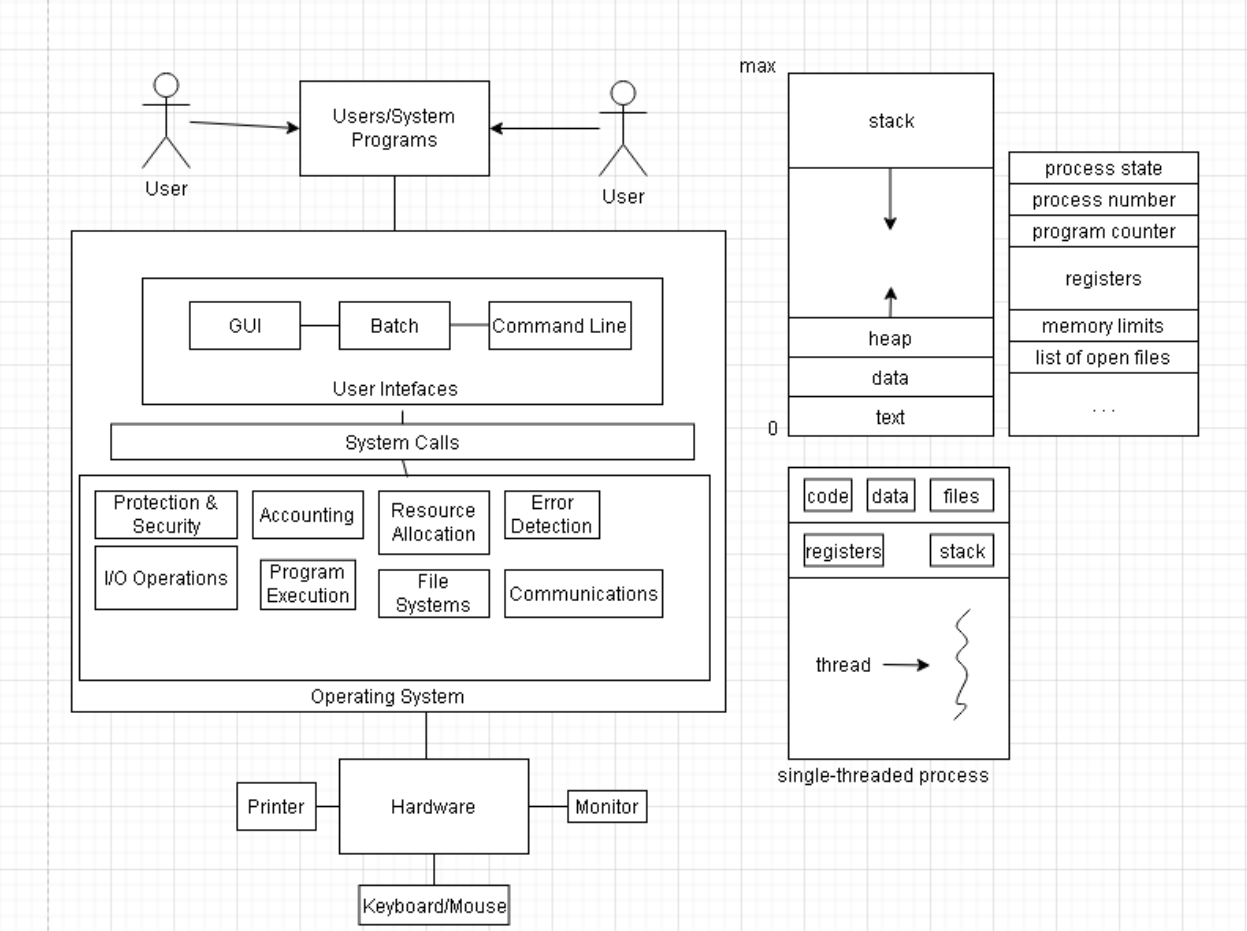
This allows each user to have a copy of the file within their own directories while having a shared one. The last directory there is to cover is the general graph directory and this one is a two -level directory that allows users to created subdirectories. I/O devices include things such as disks, tapes, Bluetooth and screens, keyboards, and mouses. For the I/O devices to communicate with the CPU must initiate the device drivers for the I/O, then the I/O controller initiates it, and it is either input ready, output complete, or error generates interrupt signal. From there the CPU receives the interrupt or transfers control to the interrupt handler. The interrupt handler then processes the data and returns the interrupt, the CPU then resumes processing of the interrupted task, this can be seen within figure 12.3 in our textbook. A device communicates with the machine via a port that allows humans to control it. Memory-mapped I/O is supported by a device-control register which is mapped into an address space in the. The integration across I/O and memory components is being pulled from the quote, “The CPU executes I/O requests using the standard data-transfer instructions to read and write the device-control registers at their mapped locations in physical memory”, (Silberschatz, et. Al., 2014).

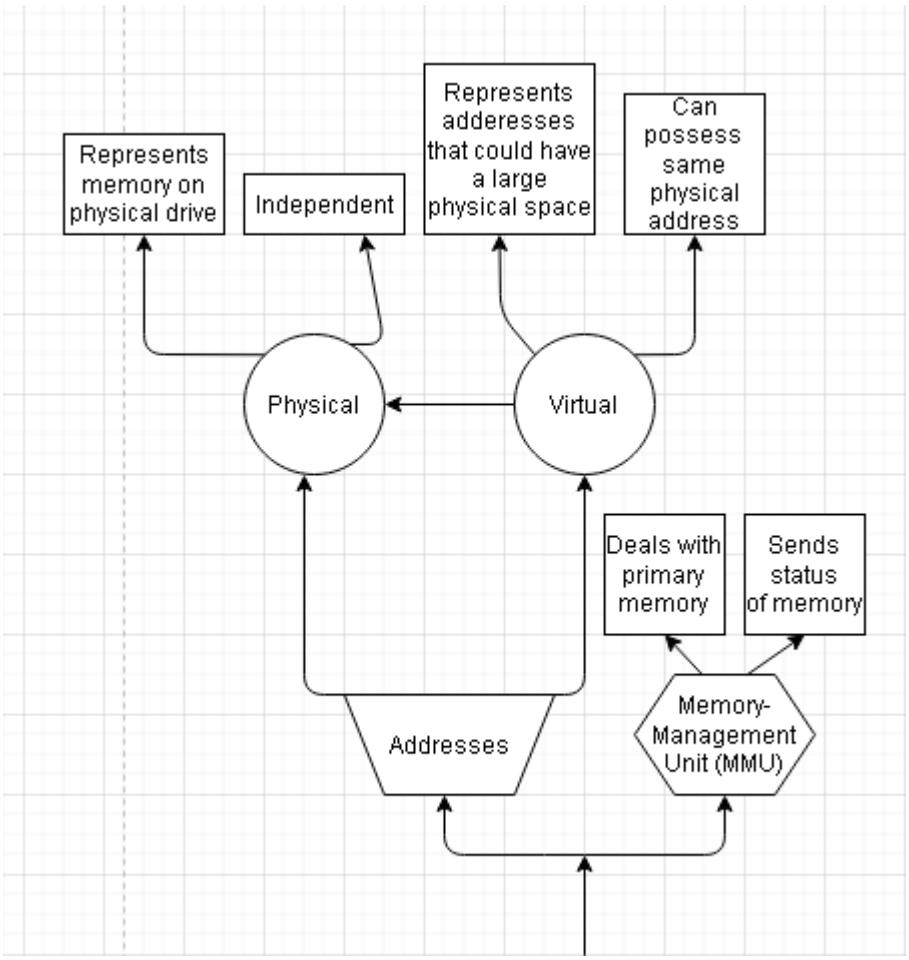
Finally, the different models of security are meant to verify who and who does not have the permissions to access rights. Language based protection is meant to utilize the kernel by simply declaring commands. Domain security models make use of access lists to give access rights to users. An access matrix can limit certain files and programs from user to user.

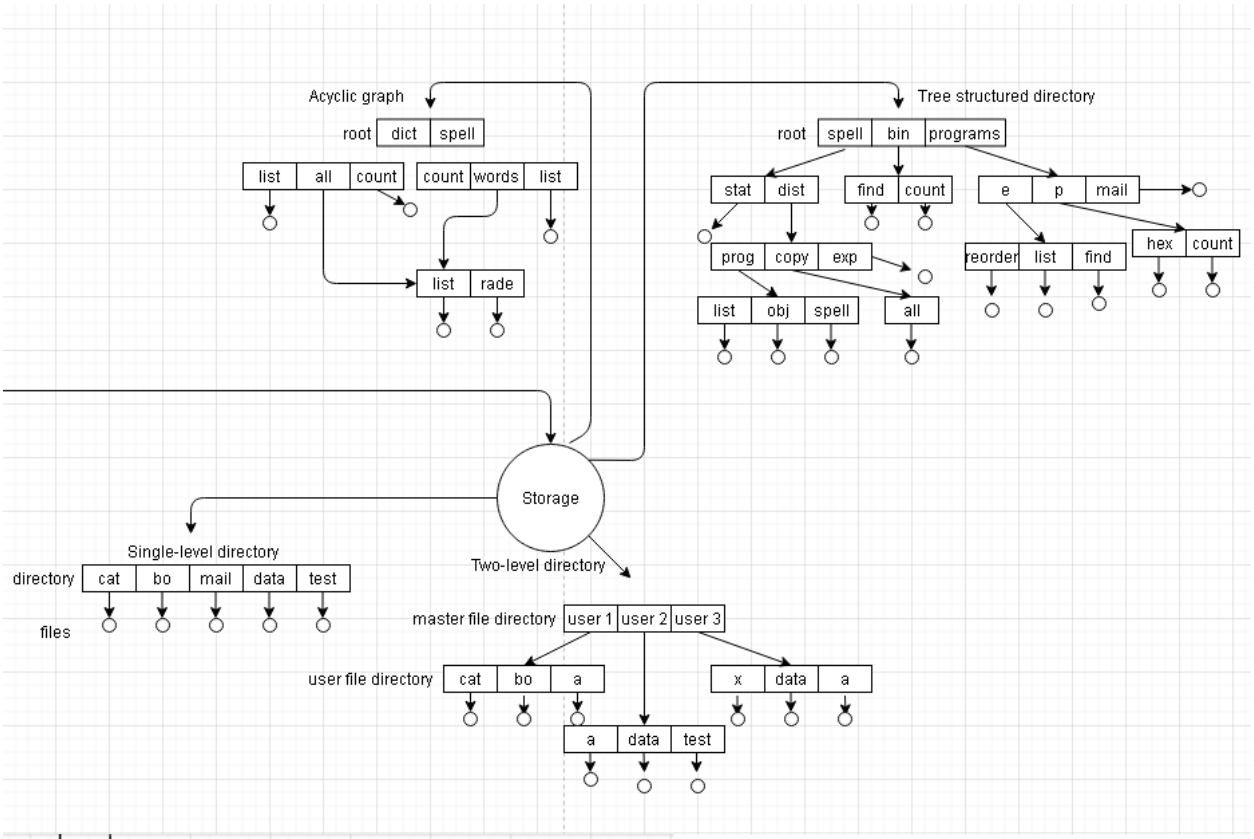
References:

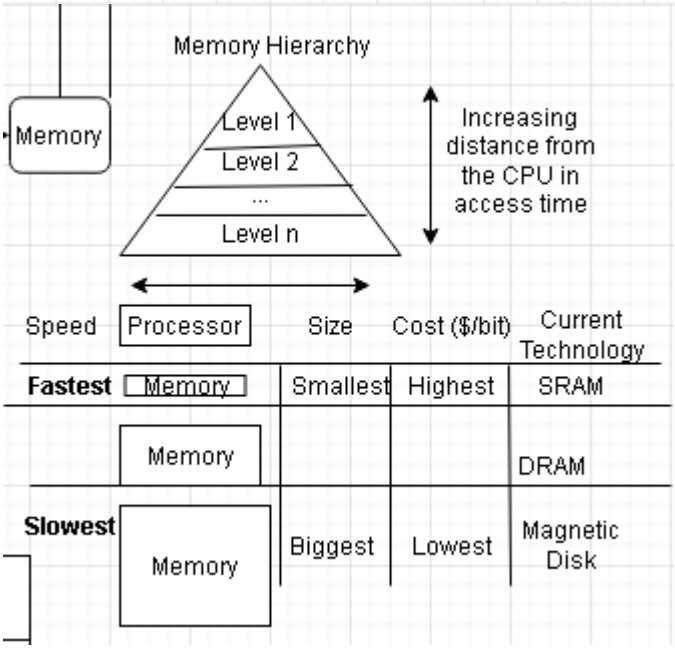
Silberschatz, A., Galvin, P. B., & Gagne, G. (2014). *Operating system concepts essentials* (2nd ed.). Retrieved from https://redshelf.com/ (Links to an external site.)

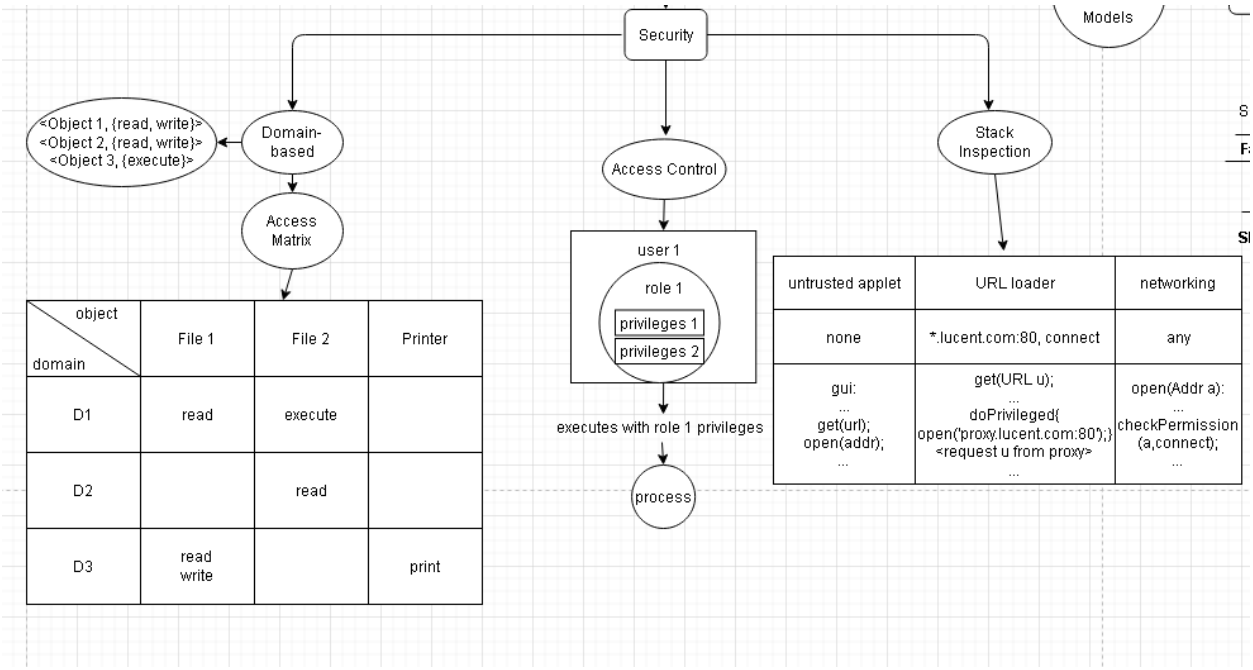
Link to concept map: https://drive.google.com/file/d/1fL24cIegdk9JS2oMj8l-YkEVdp7JuA5S/view?usp=sharing




Represents memory on physical drive
Independent
Represents adderesses that could have a large physical space
Can possess same physical address
Physical
Virtual
Deals with primary memory
Sends status of memory
Addresses
Memory-Management Unit (MMU)


Acyclic graph
root
dict
spell
list
all
count
count
words
list
list
rade
Tree structured directory
root
spell
bin
programs
stat
dist
find
count
e
p
mail
prog
copy
exp
reorder
list
find
hex
count
list
obj
spell
all
Storage
Single-level directory
directory
cat
bo
mail
data
test
files
Two-level directory
master file directory
user 1
user 2
user 3
user file directory
cat
bo
a
x
data
a
a
data
test


Memory Hierarchy
Memory
Level 1
Level 2
...
Level n
Increasing distance from the CPU in access time
Speed
Processor
Size
Cost ($/bit)
Current Technology
Fastest
Memory
Smallest
Highest
SRAM
Memory
DRAM
Slowest
Memory
Biggest
Lowest
Magnetic Disk


Models
Security
Domain-based
<Object 1, {read, write}>
<Object 2, {read, write}>
<Object 3, {execute}>
Access Matrix
Access Control
Stack Inspection
user 1
role 1
privileges 1
privileges 2
executes with role 1 privileges
process
object
domain
File 1
File 2
Printer
D1
read
execute
D2
read
D3
read
write
print
untrusted applet
URL loader
networking
none
*.lucent.com:80, connect
any
gui:
...
get(url);
open(addr);
...
get(URL u);
...
doPrivileged(
open('proxy.lucent.com:80');}
<request u from proxy>
...
open(Addr a):
...
checkPermission
(a,connect);
...