Homework7

Name Yu Yang   Student ID 892449550

14.1

Each node in access lists represents an object's domains and access rights.
While each node in capability lists represents a domain's allowed access operations associated with objects.

14.10

Even the capability list is associated with a domain, but it is never directly accessible to a process executing in that domain. Rather, the capability list is itself a protected object, maintained by the operating system and accessed by the user only indirectly.

14.15

The strength is the access rights is with the object itself, set we could remove or add access rights in the list conveniently.
The weakness is to check a domain's access rights on the object, we need to iterate the whole list (in worst case), which is very expensive but need to be done every time we access the object.

14. 16

The strength is whenever an object is accessed, the system only needs to check the capabilities matches what are in the domain's capabilities, this is much faster than the one associated with objects. And capability could be copy to other domains easily.
The weakness is the lack of control: it is more difficult when we want to remove or change the capabilities.

15.3

Salt is a random number which is generated by system and added to the password. The "salted" password is encrypted and stored with salt in a password file.
When system performs password check, the password provided by user is append with salt and encrypted, then compare with the stored one. Hacker couldn't use a single password to compare with all the passwords encrypted simultaneously because salt are different for different users.

15.6

The COPS itself could be hacked and disabled.
Or itself could have security issues that could not be scanned.
To solve it, we could tighten the access to COPS, only system manager above certain level could get the COPS, and we could encrypt the COPS before store it on media.

15.9

Physical sites where computer systems are stored: human and physical
Network breach: human, OS and physical
Modem breach: human and physical
Malicious log in: OS and human
Backup media protection: human and physical
Programmers for the system: human

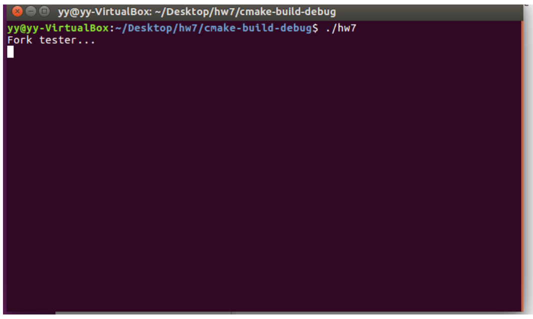
15.14

a. Use private key of the sender to encode message
b. Use public key of the receiver to encode message
c. Using both keys mention of a & b in encryption.

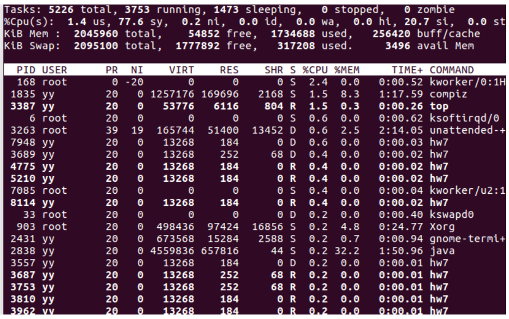
VM

Screenshot

Fork bomb



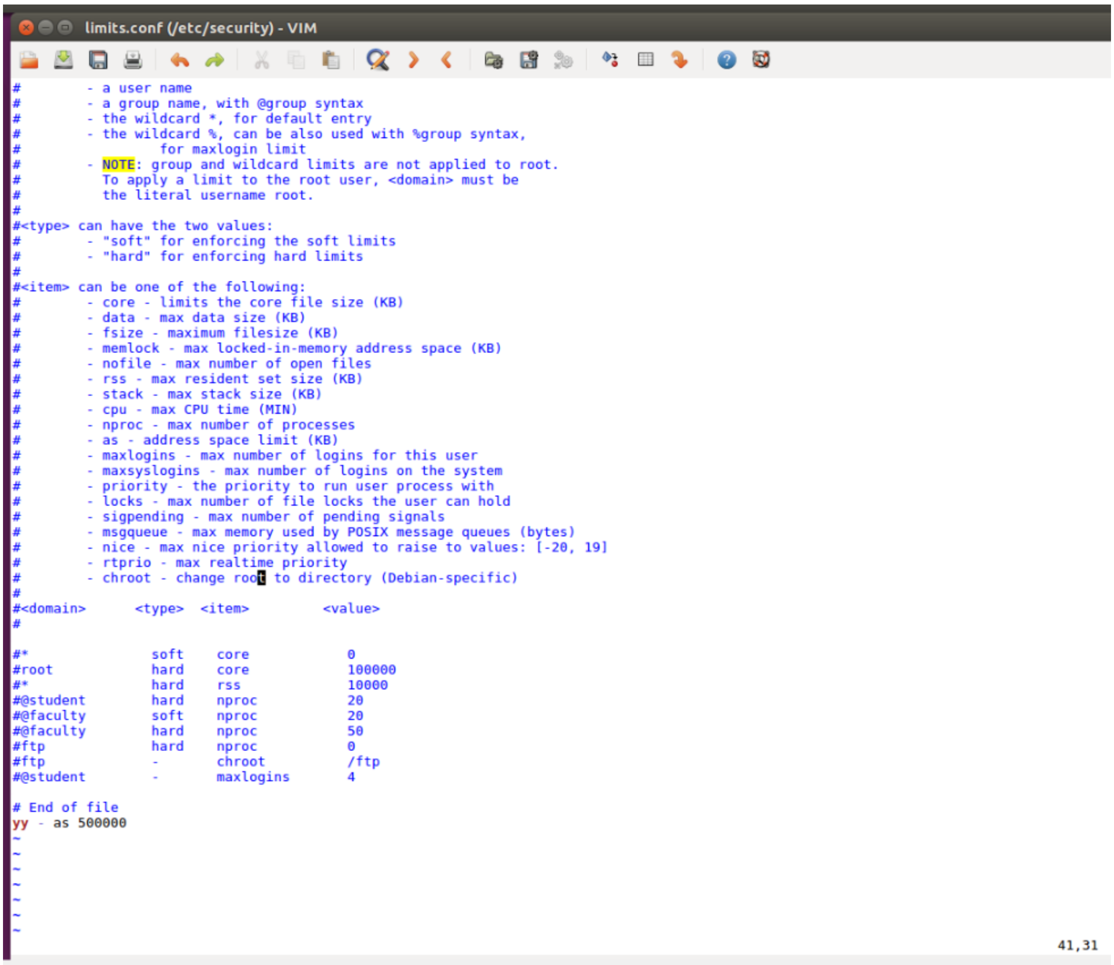
```
yy@yy-VirtualBox:~/Desktop/hw7/cmake-build-debug$ ./hw7
Fork tester...
```



```
Tasks: 5226 total, 3753 running, 1473 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.4 us, 77.6 sy,  0.2 ni,  0.0 id,  0.0 wa,  0.0 hi, 20.7 si,  0.0 st
KiB Mem :  2045960 total,    54852 free,  1734688 used,   256420 buff/cache
KiB Swap:  2095100 total,  1777892 free,   317208 used.     3496 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  168 root       0 -20       0      0      0 S  2.4  0.0   0:00.52 kworker/0:1H
 1835 yy        20   0 1257176 169696   2168 S  1.5  8.3   1:17.59 compiz
 3387 yy        20   0   53776   6116    804 R  1.5  0.3   0:00.26 top
    6 root      20   0       0      0      0 S  0.6  0.0   0:00.62 ksoftirqd/0
 3263 root      39  19  165744  51400  13452 D  0.6  2.5   2:14.05 unattended-+
 7948 yy        20   0   13268    184      0 D  0.6  0.0   0:00.03 hw7
 3689 yy        20   0   13268    252     68 D  0.4  0.0   0:00.02 hw7
 4775 yy        20   0   13268    184      0 R  0.4  0.0   0:00.02 hw7
 5210 yy        20   0   13268    184      0 R  0.4  0.0   0:00.02 hw7
 7085 root      20   0       0      0      0 S  0.4  0.0   0:00.04 kworker/u2:1
 8114 yy        20   0   13268    184      0 R  0.4  0.0   0:00.02 hw7
   33 root      20   0       0      0      0 D  0.2  0.0   0:00.40 kswapd0
  903 root      20   0  498436  97424  16856 S  0.2  4.8   0:24.77 Xorg
 2431 yy        20   0  673568  15284   2588 S  0.2  0.7   0:00.94 gnome-termi+
 2838 yy        20   0 4559836 657816     44 S  0.2 32.2   1:50.96 java
 3557 yy        20   0   13268    184      0 D  0.2  0.0   0:00.01 hw7
 3687 yy        20   0   13268    252     68 R  0.2  0.0   0:00.01 hw7
 3753 yy        20   0   13268    252     68 R  0.2  0.0   0:00.01 hw7
 3810 yy        20   0   13268    184      0 R  0.2  0.0   0:00.01 hw7
 3962 yy        20   0   13268    184      0 R  0.2  0.0   0:00.01 hw7
```

Memory limit



```
#        - a user name
#        - a group name, with @group syntax
#        - the wildcard *, for default entry
#        - the wildcard %, can be also used with %group syntax,
#                 for maxlogin limit
#        - NOTE: group and wildcard limits are not applied to root.
#          To apply a limit to the root user, <domain> must be
#          the literal username root.
#
#<type> can have the two values:
#        - "soft" for enforcing the soft limits
#        - "hard" for enforcing hard limits
#
#<item> can be one of the following:
#        - core - limits the core file size (KB)
#        - data - max data size (KB)
#        - fsize - maximum filesize (KB)
#        - memlock - max locked-in-memory address space (KB)
#        - nofile - max number of open files
#        - rss - max resident set size (KB)
#        - stack - max stack size (KB)
#        - cpu - max CPU time (MIN)
#        - nproc - max number of processes
#        - as - address space limit (KB)
#        - maxlogins - max number of logins for this user
#        - maxsyslogins - max number of logins on the system
#        - priority - the priority to run user process with
#        - locks - max number of file locks the user can hold
#        - sigpending - max number of pending signals
#        - msgqueue - max memory used by POSIX message queues (bytes)
#        - nice - max nice priority allowed to raise to values: [-20, 19]
#        - rtprio - max realtime priority
#        - chroot - change root to directory (Debian-specific)
#
#<domain>      <type>  <item>         <value>
#

#*               soft    core            0
#root            hard    core            100000
#*               hard    rss             10000
#@student        hard    nproc           20
#@faculty        soft    nproc           20
#@faculty        hard    nproc           50
#ftp             hard    nproc           0
#ftp             -       chroot          /ftp
#@student        -       maxlogins       4

# End of file
yy - as 500000
~
~
~
~
~
~
~
                                                                        41,31
```

Questions

1.

```
-p
Use pipe() instead of socketpair() for inter-process
communication
-t
Use multi thread instead of multi process
-g
Specify the number of groups
```

2.

Without p, socketpair is used and test time increased.
Reason: pipes reger to buffering between virtual files, they don't use packets which sockets ue. And sockets is based on IPv4 and IPv6. So sockets are more time-consuming.

3.

Without t, multi processes are used instead of threads and test time increased.
Reason: because a thread is only an entity in a thread and thread is light-weighted than process.