Lab08 Virtual Memory in Minix3

Hu Zhengdong, 517370910249

2020, Nov. 22th

2.1. Memory management at kernel level

- What does vm stands for?
  Virtual Memory. VM is used when we need to run one or several programs using large amount of memory, which helps the program to complete.

- Find all the place where the vm used inside the kernel
  grep -r vm /usr/src/kernel

- How is memory allocated within the kernel? Why are not malloc and calloc used?
  Kmalloc, kmem_cache_alloc, vmalloc. Linux provides a variety of APIs for memory allocation. You can allocate small chunks using kmalloc or kmem_cache_alloc families, large virtually contiguous areas using vmalloc and its derivatives, or you can directly request pages from the page allocator with alloc_pages. It is also possible to use more specialized allocators, for instance cma_alloc or zs_malloc
  Malloc and calloc are not used because they are not defined in the kernel

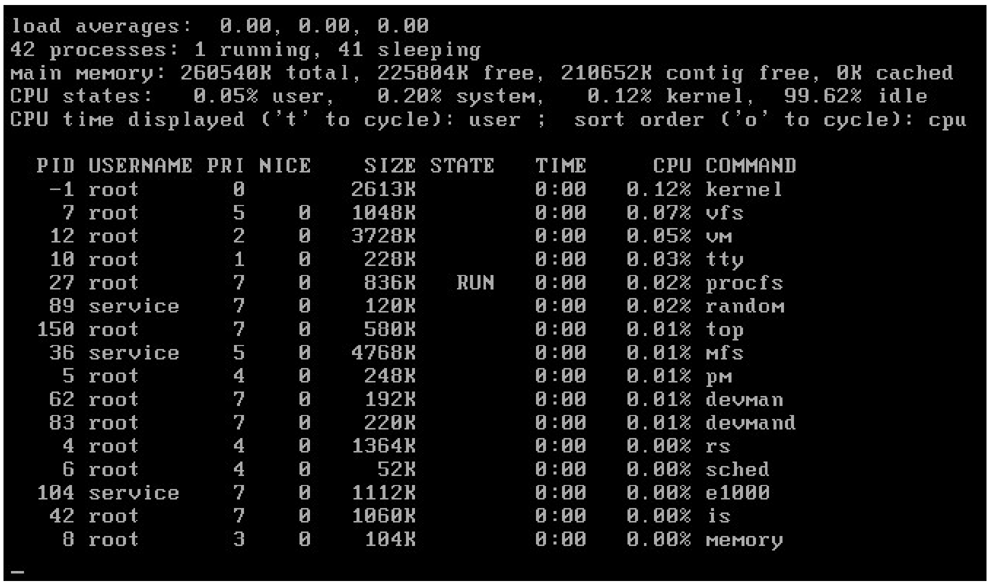
- While allocating memory, how does functions in kernel space switch back and from between user and kernel space? How is that boundary crossed? How good or bad it is to put vm in userspace?
  Message passing can be used to switch back and forth between user and kernel space. It's bad to vm in userspave which can easily cause page faults.

- How are page faults handled?
  When handling a page fault, the operating system tries to make the required page accessible at the location in physical memory or terminates the program in cases of an illegal memory access. Actually there are two kinds of page faults. One is that the virtual address is illegal, which can't be found in virtual address space. Another is because the page is removed from the physical memory to disk.

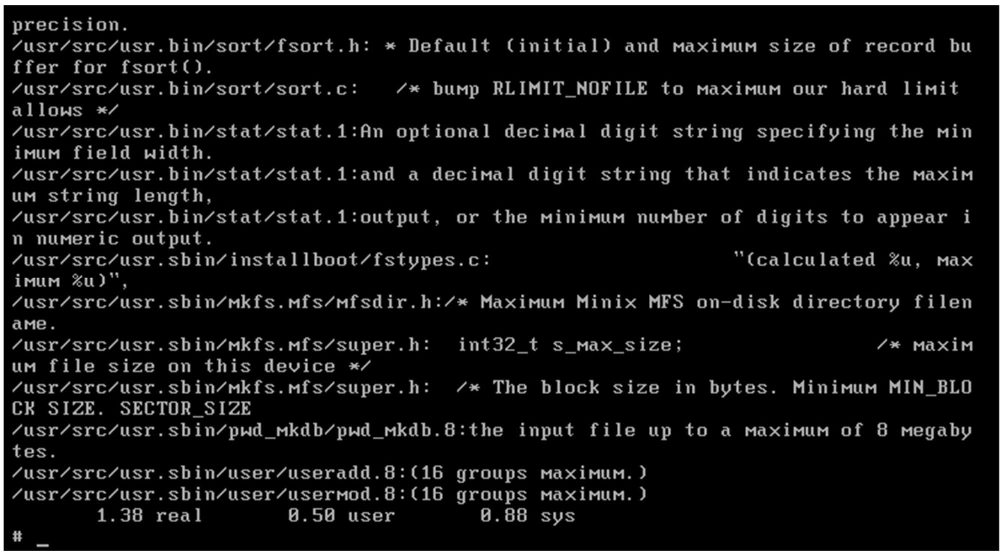
2.2. Mum's Really Unfair

- What algorithm is used by default in Minix 3 to handle pagefault? Find its implementation and study it closely.
  The algorithm used by default in Minix3 is LRU method
  grep -r lru /usr/src/servers/vm

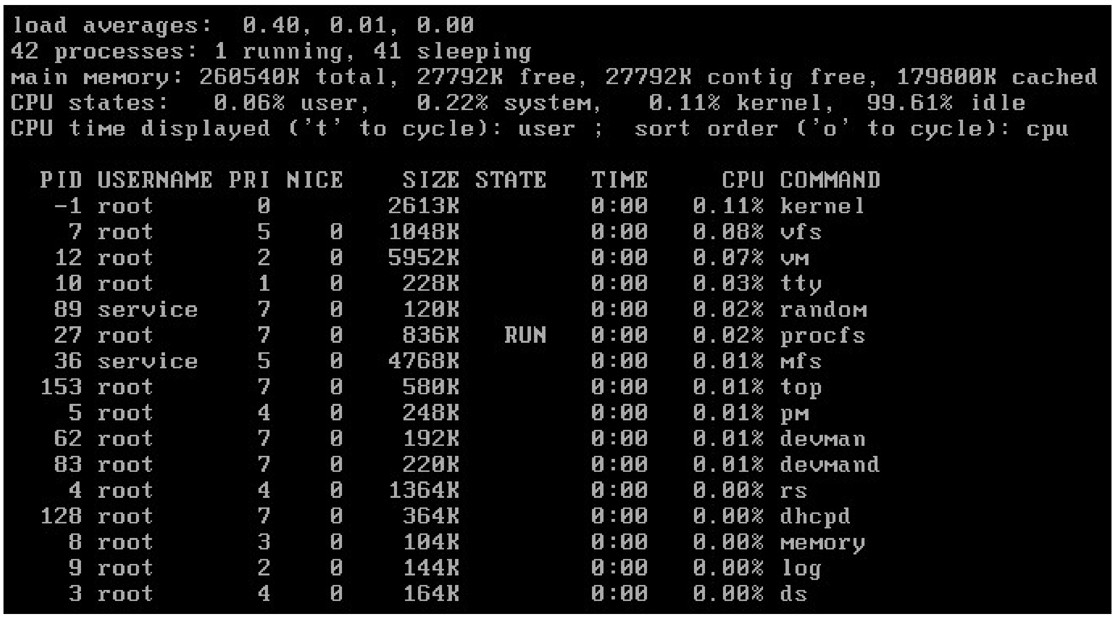
- Use the top command to keep track of your used memory and cache, then runtime grep -r "mum" /usr/src. Run the command again. What do you notice?



```
load averages:  0.00, 0.00, 0.00
42 processes: 1 running, 41 sleeping
main memory: 260540K total, 225804K free, 210652K contig free, 0K cached
CPU states:   0.05% user,   0.20% system,   0.12% kernel,  99.62% idle
CPU time displayed ('t' to cycle): user ;  sort order ('o' to cycle): cpu

  PID USERNAME PRI NICE    SIZE STATE   TIME     CPU COMMAND
   -1 root       0        2613K         0:00   0.12% kernel
    7 root       5    0   1048K         0:00   0.07% vfs
   12 root       2    0   3728K         0:00   0.05% vm
   10 root       1    0    228K         0:00   0.03% tty
   27 root       7    0    836K   RUN   0:00   0.02% procfs
   89 service    7    0    120K         0:00   0.02% random
  150 root       7    0    580K         0:00   0.01% top
   36 service    5    0   4768K         0:00   0.01% mfs
    5 root       4    0    248K         0:00   0.01% pm
   62 root       7    0    192K         0:00   0.01% devman
   83 root       7    0    220K         0:00   0.01% devmand
    4 root       4    0   1364K         0:00   0.00% rs
    6 root       4    0     52K         0:00   0.00% sched
  104 service    7    0   1112K         0:00   0.00% e1000
   42 root       7    0   1060K         0:00   0.00% is
    8 root       3    0    104K         0:00   0.00% memory
_
```



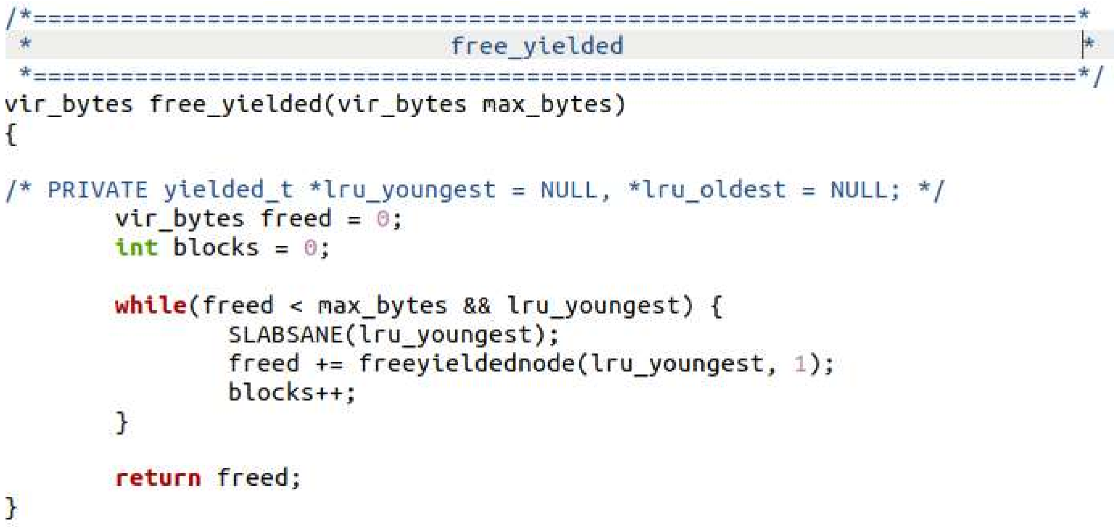
```
precision.
/usr/src/usr.bin/sort/fsort.h: * Default (initial) and maximum size of record bu
ffer for fsort().
/usr/src/usr.bin/sort/sort.c:   /* bump RLIMIT_NOFILE to maximum our hard limit
allows */
/usr/src/usr.bin/stat/stat.1:An optional decimal digit string specifying the min
imum field width.
/usr/src/usr.bin/stat/stat.1:and a decimal digit string that indicates the maxim
um string length,
/usr/src/usr.bin/stat/stat.1:output, or the minimum number of digits to appear i
n numeric output.
/usr/src/usr.sbin/installboot/fstypes.c:                    "(calculated %u, max
imum %u)",
/usr/src/usr.sbin/mkfs.mfs/mfsdir.h:/* Maximum Minix MFS on-disk directory filen
ame.
/usr/src/usr.sbin/mkfs.mfs/super.h:  int32_t s_max_size;              /* maxim
um file size on this device */
/usr/src/usr.sbin/mkfs.mfs/super.h:  /* The block size in bytes. Minimum MIN_BLO
CK SIZE. SECTOR_SIZE
/usr/src/usr.sbin/pwd_mkdb/pwd_mkdb.8:the input file up to a maximum of 8 megaby
tes.
/usr/src/usr.sbin/user/useradd.8:(16 groups maximum.)
/usr/src/usr.sbin/user/usermod.8:(16 groups maximum.)
       1.38 real       0.50 user       0.88 sys
# _
```



```
load averages:  0.40, 0.01, 0.00
42 processes: 1 running, 41 sleeping
main memory: 260540K total, 27792K free, 27792K contig free, 179800K cached
CPU states:   0.06% user,   0.22% system,   0.11% kernel,  99.61% idle
CPU time displayed ('t' to cycle): user ;  sort order ('o' to cycle): cpu

  PID USERNAME PRI NICE    SIZE STATE   TIME     CPU COMMAND
   -1 root       0        2613K         0:00   0.11% kernel
    7 root       5    0   1048K         0:00   0.08% vfs
   12 root       2    0   5952K         0:00   0.07% vm
   10 root       1    0    228K         0:00   0.03% tty
   89 service    7    0    120K         0:00   0.02% random
   27 root       7    0    836K   RUN   0:00   0.02% procfs
   36 service    5    0   4768K         0:00   0.01% mfs
  153 root       7    0    580K         0:00   0.01% top
    5 root       4    0    248K         0:00   0.01% pm
   62 root       7    0    192K         0:00   0.01% devman
   83 root       7    0    220K         0:00   0.01% devmand
    4 root       4    0   1364K         0:00   0.00% rs
  128 root       7    0    364K         0:00   0.00% dhcpd
    8 root       3    0    104K         0:00   0.00% memory
    9 root       2    0    144K         0:00   0.00% log
    3 root       4    0    164K         0:00   0.00% ds
```

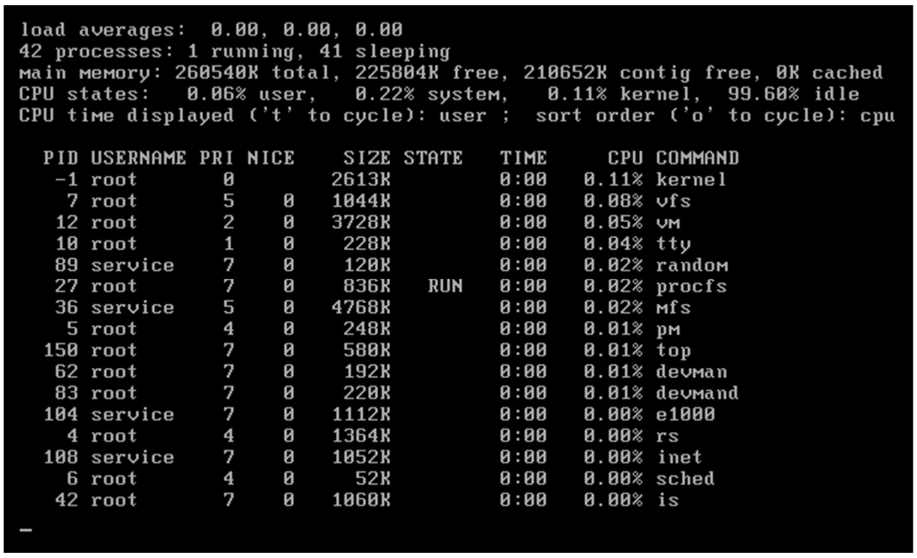
The cached memory increased from 0K to 179800K.
The time is about 1.38 (real), 0.50 (user), 0.88 (sys).

- Adjust the implementation of LRU into MRU and recompile the kernel.
  We can change the freed node from lru_oldest to lru_youngest

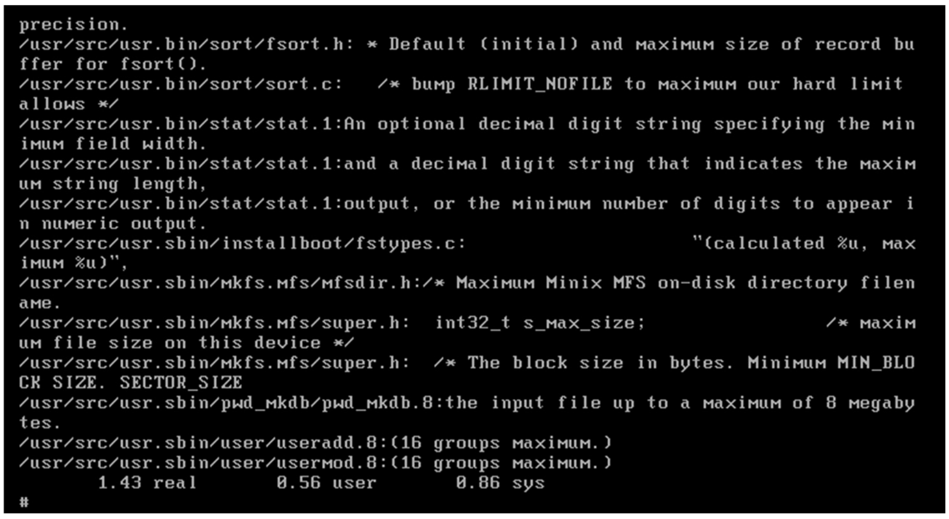


```
/*===========================================================================*
 *                              free_yielded                                 *
 *===========================================================================*/
vir_bytes free_yielded(vir_bytes max_bytes)
{

/* PRIVATE yielded_t *lru_youngest = NULL, *lru_oldest = NULL; */
	vir_bytes freed = 0;
	int blocks = 0;

	while(freed < max_bytes && lru_youngest) {
		SLABSANE(lru_youngest);
		freed += freeyieldednode(lru_youngest, 1);
		blocks++;
	}

	return freed;
}
```

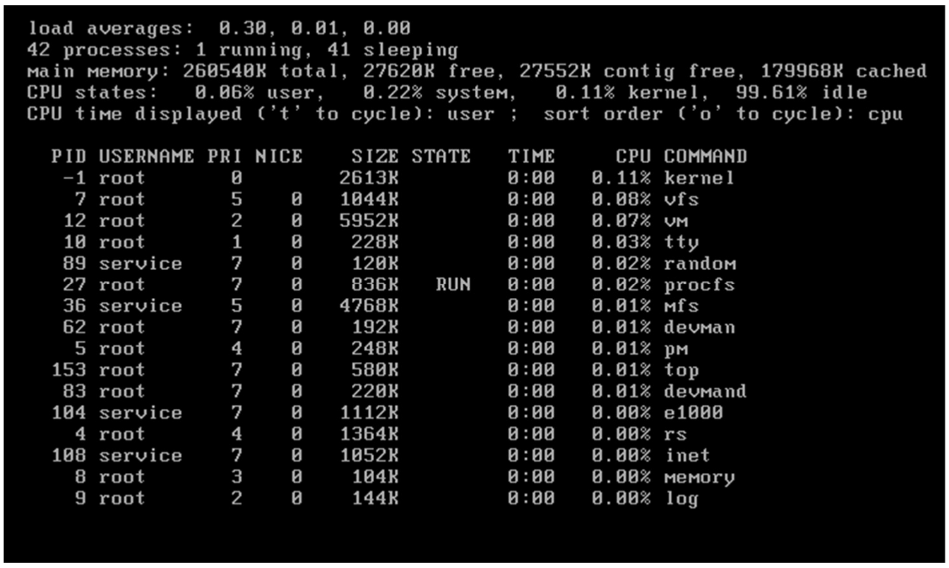
- Use the top command to keep track of your used memory and cache, then runtime grep -r "mum" /usr/src. Run the command again. What do you notice?



```
load averages:  0.00, 0.00, 0.00
42 processes: 1 running, 41 sleeping
main memory: 260540K total, 225804K free, 210652K contig free, 0K cached
CPU states:   0.06% user,   0.22% system,   0.11% kernel,  99.60% idle
CPU time displayed ('t' to cycle): user ;  sort order ('o' to cycle): cpu

  PID USERNAME PRI NICE    SIZE STATE    TIME     CPU COMMAND
   -1 root       0        2613K          0:00   0.11% kernel
    7 root       5    0   1044K          0:00   0.08% vfs
   12 root       2    0   3728K          0:00   0.05% vm
   10 root       1    0    228K          0:00   0.04% tty
   89 service    7    0    120K          0:00   0.02% random
   27 root       7    0    836K   RUN    0:00   0.02% procfs
   36 service    5    0   4768K          0:00   0.02% mfs
    5 root       4    0    248K          0:00   0.01% pm
  150 root       7    0    580K          0:00   0.01% top
   62 root       7    0    192K          0:00   0.01% devman
   83 root       7    0    220K          0:00   0.01% devmand
  104 service    7    0   1112K          0:00   0.00% e1000
    4 root       4    0   1364K          0:00   0.00% rs
  108 service    7    0   1052K          0:00   0.00% inet
    6 root       4    0     52K          0:00   0.00% sched
   42 root       7    0   1060K          0:00   0.00% is
_
```



```
precision.
/usr/src/usr.bin/sort/fsort.h: * Default (initial) and maximum size of record bu
ffer for fsort().
/usr/src/usr.bin/sort/sort.c:   /* bump RLIMIT_NOFILE to maximum our hard limit
allows */
/usr/src/usr.bin/stat/stat.1:An optional decimal digit string specifying the min
imum field width.
/usr/src/usr.bin/stat/stat.1:and a decimal digit string that indicates the maxim
um string length,
/usr/src/usr.bin/stat/stat.1:output, or the minimum number of digits to appear i
n numeric output.
/usr/src/usr.sbin/installboot/fstypes.c:                        "(calculated %u, max
imum %u)",
/usr/src/usr.sbin/mkfs.mfs/mfsdir.h:/* Maximum Minix MFS on-disk directory filen
ame.
/usr/src/usr.sbin/mkfs.mfs/super.h:  int32_t s_max_size;              /* maxim
um file size on this device */
/usr/src/usr.sbin/mkfs.mfs/super.h:  /* The block size in bytes. Minimum MIN_BLO
CK SIZE. SECTOR_SIZE
/usr/src/usr.sbin/pwd_mkdb/pwd_mkdb.8:the input file up to a maximum of 8 megaby
tes.
/usr/src/usr.sbin/user/useradd.8:(16 groups maximum.)
/usr/src/usr.sbin/user/usermod.8:(16 groups maximum.)
        1.43 real       0.56 user       0.86 sys
#
```



```
load averages:  0.30, 0.01, 0.00
42 processes: 1 running, 41 sleeping
main memory: 260540K total, 27620K free, 27552K contig free, 179968K cached
CPU states:   0.06% user,   0.22% system,   0.11% kernel,  99.61% idle
CPU time displayed ('t' to cycle): user ;  sort order ('o' to cycle): cpu

  PID USERNAME PRI NICE    SIZE STATE    TIME     CPU COMMAND
   -1 root       0        2613K          0:00   0.11% kernel
    7 root       5    0   1044K          0:00   0.08% vfs
   12 root       2    0   5952K          0:00   0.07% vm
   10 root       1    0    228K          0:00   0.03% tty
   89 service    7    0    120K          0:00   0.02% random
   27 root       7    0    836K   RUN    0:00   0.02% procfs
   36 service    5    0   4768K          0:00   0.01% mfs
   62 root       7    0    192K          0:00   0.01% devman
    5 root       4    0    248K          0:00   0.01% pm
  153 root       7    0    580K          0:00   0.01% top
   83 root       7    0    220K          0:00   0.01% devmand
  104 service    7    0   1112K          0:00   0.00% e1000
    4 root       4    0   1364K          0:00   0.00% rs
  108 service    7    0   1052K          0:00   0.00% inet
    8 root       3    0    104K          0:00   0.00% memory
    9 root       2    0    144K          0:00   0.00% log
```

The cached memory increased from 0K to 179968K.

The time is about 1.43 (real), 0.56 (user), 0.86 (sys).

- Discuss the different behaviours of LRU and MRU as well as the consequences for the users. Can you think of any situation where MRU would be better than LRU?

  We can see that cached memory after the grep command is larger for MRU than LRU, and LRU is faster than MRU to complete the process. MRU can be better than LRU when the lru_oldest in the virtual memory is frequently used.