#uni/semester3/Betriebssysteme/chapter13/code-fragen

1. The first Linux tool you should check out is the very simple tool free. First, type man free and read its entire manual page; it's short, don't worry!

2. Now, run free, perhaps using some of the arguments that might be useful (e.g., -m, to display memory totals in megabytes). How much memory is in your system? How much is free? Do these numbers match your intuition?

```
vl161bra@ct-bsys-ws20-12:~$ free -m -w
               total        used        free      shared     buffers       cache   available
Mem:            2048         328        1589          17           0         130        1719
Swap:           3072         212        2859
```

   Free Command in Linux | Linuxize
   Wie viel Speicher ist in System? 2048 + 3072 = 5120 Megabyte
   Wie viel ist frei? 3105 MB
   used = total - free - buffers - cache
   Passt es zur deiner Intuition? Ja, es scheint nach einem realistischen Speicher
   Available ist Speicher fürs starten von neuen Applikationen das verfügbar ist, ohne swapping
   Buffers wird von kernel buffers verwendet.
   Cache wird von page cache and slabs verwendet.

3. Next, create a little program that uses a certain amount of memory, called memory-user.c. This program should take one commandline argument: the number of megabytes of memory it will use. When run, it should allocate an array, and constantly stream through the array, touching each entry. The program should do this indefinitely, or, perhaps, for a certain amount of time also specified at the command line.

4. Now, while running your memory-user program, also (in a different terminal window, but on the same machine) run the free tool. How do the memory usage totals change when your program is running? How about when you kill the memory-user program? Do the numbers match your expectations? Try this for different amounts of memory usage. What happens when you use really large amounts of memory?

   Versuch 1:
   ➜ Beobachtungen nach dem ersten Laufen:

```
./memory-user 1000 100000
```

Davor!

```
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         725        1029          67         292        1322
Swap:           3072           1        3070
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         730        1024          67         292        1317
Swap:           3072           1        3070
```

Ersten 4

```
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048        1675          79          67         292         372
Swap:           3072           2        3069
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048        1728          26          67         292         319
Swap:           3072           1        3070
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048        1728          26          67         292         319
Swap:           3072           1        3070
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048        1728          26          67         292         319
Swap:           3072           1        3070
```

Letzten 4

```
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048        1727          27          67         292         320
Swap:           3072           1        3070
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048        1727          27          67         292         320
Swap:           3072           1        3070
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048        1730          24          67         292         317
Swap:           3072           1        3070
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048        1727          27          67         292         320
Swap:           3072           1        3070
```

- ➜ Die Spalten total, shared und buff/cache haben sich gar nicht geändert!
- ➜ Memory used wird um 1000 gesteigert.
- ➜ Memory free wird um 1000 verkleinert
- ➜ available wird um 1000 verkleinert

Versuch 2:

```
vl161bra@ct-bsys-ws20-12:~$ top



top - 07:34:09 up 22 days, 20:38,  2 users,  load average: 3,08, 2,98, 3,01
Tasks:  62 total,   2 running,  60 sleeping,   0 stopped,   0 zombie
%Cpu(s): 23,5 us,  0,0 sy,  0,0 ni, 76,5 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   2048,0 total,     22,6 free,   1738,0 used,    287,3 buff/cache
MiB Swap:   3072,0 total,   3070,0 free,      2,0 used.    310,0 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
26727 vl161bra  20   0 1026276   1,0g   1164 R  91,7  48,9   0:22.46 memory-user
    1 root      20   0  169520  10348   8036 S   0,0   0,5   0:08.81 systemd
   52 root      20   0   74844  48180  46804 S   0,0   2,3   0:17.00 systemd-journal
   75 root      20   0    9840   5988   4324 S   0,0   0,3   0:06.32 dhclient
   77 root      20   0    5508   2296   2084 S   0,0   0,1   0:02.59 cron
   78 root      20   0  239124   8504   7400 S   0,0   0,4   0:26.16 accounts-daemon
   79 root      20   0  225820   3784   3008 S   0,0   0,2   0:03.30 rsyslogd
   80 root      20   0   19508   7136   6168 S   0,0   0,3   0:02.75 systemd-logind
```

Memory vor dem KIll:

```
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048        1759           1          68         287         288
Swap:           3072           1        3070
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048        1737          22          68         287         310
Swap:           3072           1        3070
```

Nachdem kill:

```
vl161bra@ct-bsys-ws20-12:~$ kill 26727
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         742        1017          68         287        1305
Swap:           3072           1        3070
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         735        1024          68         287        1312
Swap:           3072           1        3070
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         735        1025          68         287        1312
Swap:           3072           1        3070
```

➜ Alles wird zurückgesetzt, also ja es entspricht meiner Erwartungen

Versuch 3:

Wenn man mehr Speicher allokieren will, als man freier Speicher hat:

```
vl161bra@ct-bsys-ws20-12:~/htwg/S3/BS/BSCode/c13_vm-intro$ ./memory-user 2500 100000
```

Speicher davor:

```
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         735        1025          67         286        1312
Swap:           3072           1        3070
```

Speicher danach und zum Ende:

```
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         735        1025          67         286        1312
Swap:           3072           1        3070
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         735        1025          67         286        1312
Swap:           3072           1        3070
```

```
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         736        1025          67         286        1312
Swap:           3072           1        3070
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         735        1025          67         286        1312
Swap:           3072           1        3070
```

Nach dem Killen:

```
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         754        1006          67         286        1293
Swap:           3072           1        3070
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         734        1026          67         286        1313
Swap:           3072           1        3070
```

Versuch 4:

```
vl161bra@ct-bsys-ws20-12:~/htwg/S3/BS/BSCode/c13_vm-intro$ ./memory-user 10000 100000
Terminated
```

Erstes ist vor dem Befehl, letzte zwei nach dem Befehl

```
vl161bra@ct-bsys-ws20-12:~$ free -m -w
               total        used        free      shared     buffers       cache   available
Mem:            2048         172        1832          17           0          43        1875
Swap:           3072         275        2796
vl161bra@ct-bsys-ws20-12:~$
vl161bra@ct-bsys-ws20-12:~$
vl161bra@ct-bsys-ws20-12:~$ free -m -w
               total        used        free      shared     buffers       cache   available
Mem:            2048        1984          10          17           0          52          63
Swap:           3072         274        2797
vl161bra@ct-bsys-ws20-12:~$ free -m -w
               total        used        free      shared     buffers       cache   available
Mem:            2048        1982          21          11           0          43          65
Swap:           3072         283        2788
```

```
vl161bra@ct-bsys-ws20-12:~$ kill 29667
vl161bra@ct-bsys-ws20-12:~$
vl161bra@ct-bsys-ws20-12:~$
vl161bra@ct-bsys-ws20-12:~$ free -m -w
               total        used        free      shared     buffers       cache   available
Mem:            2048         171        1833          10           0          42        1876
Swap:           3072         283        2788
```

Versuch 5:

```
vl161bra@ct-bsys-ws20-12:~/htwg/S3/BS/BSCode/c13_vm-intro$ ./memory-user 2000 100000
^C
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         330        1586          17         131        1717
Swap:           3072         212        2859
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048        1915           1          17         131         132
Swap:           3072         212        2859
```

1. Vor dem Programm
2. Während dem Programm

Versuch 6:

```
vl161bra@ct-bsys-ws20-12:~/htwg/S3/BS/BSCode/c13_vm-intro$ ./memory-user 2500 100000
^C
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         239        1736           9          71        1808
Swap:           3072         227        2844
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         231        1744           9          72        1816
Swap:           3072         227        2844
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         231        1744          10          72        1816
Swap:           3072         227        2844
```

1. Vor dem Program
2. + 3. während dem Programm

Versuch 7:

```
vl161bra@ct-bsys-ws20-12:~/htwg/S3/BS/BSCode/c13_vm-intro$ ./memory-user 5000 100000
^C
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048         231        1744          10          72        1816
Swap:           3072         227        2844
vl161bra@ct-bsys-ws20-12:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:            2048        1993           0          10          54          54
Swap:           3072         286        2785
```

Versuch 8:

```
vl161bra@ct-bsys-ws20-12:~/htwg/S3/BS/BSCode/c13_vm-intro$ ./memory-user 10000 100000
^C
```

```
vl161bra@ct-bsys-ws20-12:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:           2048         233        1752          10          62        1814
Swap:          3072         227        2844
vl161bra@ct-bsys-ws20-12:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:           2048        1776         209          10          62         271
Swap:          3072         227        2844
vl161bra@ct-bsys-ws20-12:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:           2048         222        1775           8          49        1825
Swap:          3072         237        2834
```

1. Vor dem Programm
2. Während dem Programm
3. Nach dem kill

   ➜ Irgendwie wird beim 2500 nichts allokiert, aber bei 1000, 2000, 5000, 10000 schon!

5. Let's try one more tool, known as pmap. Spend some time, and read the pmap manual page in detail.
   https://docs.oracle.com/cd/E19253-01/816-5165/pmap-1/index.html#:~:text=for%20more%20information.-,Usage,address%20space%20of%20a%20process.&text=By%20default%2C%20pmap%20displays%20the,mapped%20object%20name%20are%20shown.

6. To use pmap, you have to know the process ID of the process you're interested in. Thus, first run ps auxw to see a list of all processes; then, pick an interesting one, such as a browser. You can also use your memory-user program in this case (indeed, you can even have that program call getpid() and print out its PID for your convenience).

```
vl161bra@ct-bsys-ws20-12:~/htwg/S3/BS/BSCode/c13_vm-intro$ ./memory-user 1000 100000
```

```
vl161bra@ct-bsys-ws20-12:~$ pmap -x 22783
22783:   ./memory-user 1000 100000
Address           Kbytes     RSS   Dirty Mode  Mapping
00005634c7632000       4       4       0 r---- memory-user
00005634c7633000       4       4       0 r-x-- memory-user
00005634c7634000       4       0       0 r---- memory-user
00005634c7635000       4       4       4 r---- memory-user
00005634c7636000       4       4       4 rw--- memory-user
00005634c773b000     132       4       4 rw---   [ anon ]
00007f5f7cc17000 1024004 1024004 1024004 rw---   [ anon ]
00007f5fbb418000     136     136       0 r---- libc-2.28.so
00007f5fbb43a000    1312     640       0 r-x-- libc-2.28.so
00007f5fbb582000     304     100       0 r---- libc-2.28.so
00007f5fbb5ce000       4       0       0 ----- libc-2.28.so
00007f5fbb5cf000      16      16      16 r---- libc-2.28.so
00007f5fbb5d3000       8       8       8 rw--- libc-2.28.so
00007f5fbb5d5000      24      20      20 rw---   [ anon ]
00007f5fbb5eb000       4       4       0 r---- ld-2.28.so
00007f5fbb5ec000     120     120       0 r-x-- ld-2.28.so
00007f5fbb60a000      32      32       0 r---- ld-2.28.so
00007f5fbb612000       4       4       4 r---- ld-2.28.so
00007f5fbb613000       4       4       4 rw--- ld-2.28.so
00007f5fbb614000       4       4       4 rw---   [ anon ]
00007ffed7950000     132      12      12 rw---   [ stack ]
00007ffed79c4000      12       0       0 r----   [ anon ]
00007ffed79c7000       4       4       0 r-x--   [ anon ]
ffffffffff600000       4       0       0 --x--   [ anon ]
---------------- ------- ------- -------
total kB         1026280 1025128 1024084
```

```
Address: start address of map
Kbytes:  size of map in kilobytes
RSS:     resident set size in kilobytes
Dirty:   dirty pages (both shared and private) in kilobytes
Mode:    permissions on map: read, write, execute, shared, private (copy on write)
Mapping: file backing the map, or '[ anon ]' for allocated memory, or '[ stack ]' for the program stack
Offset:  offset into the file
Device:  device name (major:minor)
```

Resident memory is the real physical memory that processes utilize in the current state. The resident physical memory is a pool of memory used, representing the SAP HANA, operating system, and other programs.

How to master memory management in SAP Hana - a simplified guide to help you get started

7. Now run pmap on some of these processes, using various flags (like -X) to reveal many details about the process. What do you see? How many different entities make up a modern address space, as opposed to our simple conception of code*stack*heap?

```
vl161bra@ct-bsys-ws20-12:~$ pmap -x 28852
28852:   vim memory-user.c
Address           Kbytes     RSS   Dirty Mode  Mapping
00005568fce2f000     164     164       0 r---- vim.basic
00005568fce58000    1924    1908       0 r-x-- vim.basic
00005568fd039000     400     156       0 r---- vim.basic
00005568fd09d000      52      52      52 r---- vim.basic
00005568fd0aa000     104     104      80 rw--- vim.basic
00005568fd0c4000      44      44      44 rw---   [ anon ]
00005568fd289000    2836    2676    2676 rw---   [ anon ]
00007f8c10337000      12      12       0 r---- libnss_files-2.28.so
00007f8c1033a000      28      28       0 r-x-- libnss_files-2.28.so
00007f8c10341000       8       8       0 r---- libnss_files-2.28.so
00007f8c10343000       4       0       0 ----- libnss_files-2.28.so
00007f8c10344000       4       4       4 r---- libnss_files-2.28.so
00007f8c10345000       4       4       4 rw--- libnss_files-2.28.so
00007f8c10346000      24       0       0 rw---   [ anon ]
00007f8c1034c000    3296     544       0 r---- locale-archive
00007f8c10684000       8       8       8 rw---   [ anon ]
00007f8c10686000      24      24       0 r---- libpthread-2.28.so
00007f8c1068c000      60      60       0 r-x-- libpthread-2.28.so
00007f8c1069b000      24       0       0 r---- libpthread-2.28.so
00007f8c106a1000       4       4       4 r---- libpthread-2.28.so
00007f8c106a2000       4       4       4 rw--- libpthread-2.28.so
00007f8c106a3000      16       4       4 rw---   [ anon ]
00007f8c106a7000       8       8       0 r---- libattr.so.1.1.2448
00007f8c106a9000      12      12       0 r-x-- libattr.so.1.1.2448
00007f8c106ac000       4       0       0 r---- libattr.so.1.1.2448
00007f8c106ad000       4       4       4 r---- libattr.so.1.1.2448
00007f8c106ae000       4       4       4 rw--- libattr.so.1.1.2448
00007f8c106af000       8       8       0 r---- libpcre.so.3.13.3
00007f8c106b1000     328      64       0 r-x-- libpcre.so.3.13.3
00007f8c10703000     120       0       0 r---- libpcre.so.3.13.3
00007f8c10721000       4       4       4 r---- libpcre.so.3.13.3
00007f8c10722000       4       4       4 rw--- libpcre.so.3.13.3
00007f8c10723000     136     136       0 r---- libc-2.28.so
00007f8c10745000    1312    1092       0 r-x-- libc-2.28.so
00007f8c1088d000     304     120       0 r---- libc-2.28.so
00007f8c108d9000       4       0       0 ----- libc-2.28.so
00007f8c108da000      16      16      16 r---- libc-2.28.so
00007f8c108de000       8       8       8 rw--- libc-2.28.so
00007f8c108e0000      24      24      24 rw---   [ anon ]
00007f8c108e6000       4       4       0 r---- libdl-2.28.so
00007f8c108e7000       4       4       0 r-x-- libdl-2.28.so
00007f8c108e8000       4       0       0 r---- libdl-2.28.so
00007f8c108e9000       4       4       4 r---- libdl-2.28.so
00007f8c108ea000       4       4       4 rw--- libdl-2.28.so
00007f8c108eb000      20      20       0 r-x-- libgpm.so.2
00007f8c108f0000    2048       0       0 ----- libgpm.so.2
00007f8c10af0000       4       4       4 r---- libgpm.so.2
00007f8c10af1000       4       4       4 rw--- libgpm.so.2
```

```
00007f8c10af2000       8       8       0 r---- libacl.so.1.1.2253
00007f8c10af4000      20      20       0 r-x-- libacl.so.1.1.2253
00007f8c10af9000       8       0       0 r---- libacl.so.1.1.2253
00007f8c10afb000       4       4       4 r---- libacl.so.1.1.2253
00007f8c10afc000       4       4       4 rw--- libacl.so.1.1.2253
00007f8c10afd000     148     128       0 r-x-- libselinux.so.1
00007f8c10b22000    2044       0       0 ----- libselinux.so.1
00007f8c10d21000       4       4       4 r---- libselinux.so.1
00007f8c10d22000       4       4       4 rw--- libselinux.so.1
00007f8c10d23000       8       4       4 rw---   [ anon ]
00007f8c10d25000      56      56       0 r---- libtinfo.so.6.1
00007f8c10d33000      56      56       0 r-x-- libtinfo.so.6.1
00007f8c10d41000      52      52       0 r---- libtinfo.so.6.1
00007f8c10d4e000      16      16      16 r---- libtinfo.so.6.1
00007f8c10d52000       4       4       4 rw--- libtinfo.so.6.1
00007f8c10d53000      52      52       0 r---- libm-2.28.so
00007f8c10d60000     636     300       0 r-x-- libm-2.28.so
00007f8c10dff000     852       0       0 r---- libm-2.28.so
00007f8c10ed4000       4       4       4 r---- libm-2.28.so
00007f8c10ed5000       4       4       4 rw--- libm-2.28.so
00007f8c10ed6000       8       8       8 rw---   [ anon ]
00007f8c10ee8000       4       4       0 r---- ld-2.28.so
00007f8c10ee9000     120     120       0 r-x-- ld-2.28.so
00007f8c10f07000      32      32       0 r---- ld-2.28.so
00007f8c10f0f000       4       4       4 r---- ld-2.28.so
00007f8c10f10000       4       4       4 rw--- ld-2.28.so
00007f8c10f11000       4       4       4 rw---   [ anon ]
00007fffcd8d4000     132      60      60 rw---   [ stack ]
00007fffcd9ef000      12       0       0 r----   [ anon ]
00007fffcd9f2000       4       4       0 r-x--   [ anon ]
ffffffffff600000       4       0       0 --x--   [ anon ]
---------------- ------- ------- -------
total kB           17752    8316    3088
```

➜ Libraries, heap, code, stack

9. Finally, let's run pmap on your memory-user program, with different amounts of used memory. What do you see here? Does the output from pmap match your expectations?

➜ [anon] vom Prozess ändert sich je nach allokierten Speicher

➜ Offene Fragen: Was ist RSS, Anonymous memory? Was ist eigentlich mapping? Warum RSS manchmal kleiner als map größe

➜ RSS gibt die Portion an Speicher, das sich in RAM befindet. Der Rest der memory kann auf dem swap space sein oder in file system.