Die Aufgaben wurden im HTWG Container erledigt

13.2

```
lu851not@ct-bsys-ss20-15:~/z-drive/5.Semester/BSYS/chap9$ free -m
              total        used        free      shared  buff/cache   available
Mem:           2048        1073         510          82         463         974
Swap:          3072         244        2827
```

Das System des HTWG-Containers hat 2GB Arbeitsspeicher. Davon sind 510MB frei. Diese Darstellung passt auch zu meiner Intuition, da handelsübliche Arbeitsspeicher mit 2 GB auch 2048MB haben. Das 510MB lediglich frei sind, ist auch nicht außergewöhnlich, da 2 GB nicht sonderlich viel sind. Als wir am Anfang ohne free gearbeitet haben, wurde der belegte Speicher zwar nach der Ausführung befreit, aber mit der Zeit wurde der benutzte Arbeitsspeicher immer mehr.

Die Flag -m gibt das ganze in mb an, -g würde es in gb angeben.

13.4

Der verwendete Arbeitsspeicher steigt an, während der freie Arbeitsspeicher sinkt, solange das Programm läuft, da logischerweise Arbeitsspeicher mit malloc reserviert und beschrieben wird. Wenn das Programm endet, wird der allokierte Speicher wieder freigegeben.

Wenn der Prozess memory-user gekillt wird mit z.B. ^C(Strg + c), dann wird der benutzte Arbeitsspeicher wieder freigegeben und der Prozess nicht weiter ausgeführt.

Ich interpretiere, dass 500 MB viel Arbeitsspeicher sind. Wenn viel Memory im Programm benutzt wird (z.B. 500MB), dann wird stetig Speicher reserviert, bis das Programm fertig ist und danach wieder freigegeben. Selbst wenn mehr Arbeitsspeicher reserviert werden soll als es im System gibt, ist dies möglich. Dies sind man anhand der Dirty Pages.

13.7

Pmap -x 22100 (Firefox)

```
00007f2777d9d000       4       4       4 rw--- libmozsandbox.so
00007f2777d9e000       8       8       8 rw---   [ anon ]
00007f2777da0000       4       4       0 r---- ld-2.28.so
00007f2777da1000     120     120       0 r-x-- ld-2.28.so
00007f2777dbf000      32      32       0 r---- ld-2.28.so
00007f2777dc7000       4       4       4 r---- ld-2.28.so
00007f2777dc8000       4       4       4 rw--- ld-2.28.so
00007f2777dc9000       4       4       4 rw---   [ anon ]
00007fff090cd000     124      80      80 rw---   [ stack ]
00007fff090ec000       8       4       4 rw---   [ anon ]
00007fff091d1000      12       0       0 r----   [ anon ]
00007fff091d4000       4       4       0 r-x--   [ anon ]
ffffffffff600000       4       0       0 --x--   [ anon ]
---------------- ------- ------- -------
total kB         2383028   69672   14756
```

Mit der Flag -x für Pmap bekommt man wie im obigen Bild folgende Prozessdetails (von links nach rechts):

- Die Adresse
- Die Größe der map in Kilobytes
- Residentset größe in Kilobytes
- dirty pages in Kilobytes
- Permissions

- filename (anon für reservierten Speicher, stack für den Programm-stack).

Der Adressraum ist aus deutlich mehr Teilen aufgebaut als unsere einfache Konzeption von Code / Stack / Heap.

13.8

```
lu851not@ct-bsys-ss20-15:~$ pmap -x 16445
16445:   ./memory-user 500 300
Address           Kbytes     RSS   Dirty Mode  Mapping
0000000000400000       4       4       0 r---- memory-user
0000000000401000       4       4       0 r-x-- memory-user
0000000000402000       4       4       0 r---- memory-user
0000000000403000       4       4       4 r---- memory-user
0000000000404000       4       4       4 rw--- memory-user
0000000001264000     132       4       4 rw---   [ anon ]
00007fea66dff000  512004  125116  125116 rw---   [ anon ]
00007fea86200000     136     136       0 r---- libc-2.28.so
00007fea86222000    1312     944       0 r-x-- libc-2.28.so
00007fea8636a000     304     152       0 r---- libc-2.28.so
00007fea863b6000       4       0       0 ----- libc-2.28.so
00007fea863b7000      16      16      16 r---- libc-2.28.so
00007fea863bb000       8       8       8 rw--- libc-2.28.so
00007fea863bd000      24      20      20 rw---   [ anon ]
00007fea863d3000       4       4       0 r---- ld-2.28.so
00007fea863d4000     120     120       0 r-x-- ld-2.28.so
00007fea863f2000      32      32       0 r---- ld-2.28.so
00007fea863fa000       4       4       4 r---- ld-2.28.so
00007fea863fb000       4       4       4 rw--- ld-2.28.so
00007fea863fc000       4       4       4 rw---   [ anon ]
00007ffc55df4000     132       8       8 rw---   [ stack ]
00007ffc55f70000      12       0       0 r----   [ anon ]
00007ffc55f73000       4       4       0 r-x--   [ anon ]
ffffffffff600000       4       0       0 --x--   [ anon ]
---------------- ------- ------- -------
total kB          514280  126596  125192
```

```
lu851not@ct-bsys-ss20-15:~$ pmap -x 17621
17621:   ./memory-user 800 300
Address           Kbytes     RSS   Dirty Mode  Mapping
0000000000400000       4       4       0 r---- memory-user
0000000000401000       4       4       0 r-x-- memory-user
0000000000402000       4       4       0 r---- memory-user
0000000000403000       4       4       4 r---- memory-user
0000000000404000       4       4       4 rw--- memory-user
00000000015e6000     132       4       4 rw---   [ anon ]
00007fe78fa87000  819204   55336   55336 rw---   [ anon ]
00007fe7c1a88000     136     136       0 r---- libc-2.28.so
00007fe7c1aaa000    1312     896       0 r-x-- libc-2.28.so
00007fe7c1bf2000     304     120       0 r---- libc-2.28.so
00007fe7c1c3e000       4       0       0 ----- libc-2.28.so
00007fe7c1c3f000      16      16      16 r---- libc-2.28.so
00007fe7c1c43000       8       8       8 rw--- libc-2.28.so
00007fe7c1c45000      24      20      20 rw---   [ anon ]
00007fe7c1c5b000       4       4       0 r---- ld-2.28.so
00007fe7c1c5c000     120     120       0 r-x-- ld-2.28.so
00007fe7c1c7a000      32      32       0 r---- ld-2.28.so
00007fe7c1c82000       4       4       4 r---- ld-2.28.so
00007fe7c1c83000       4       4       4 rw--- ld-2.28.so
00007fe7c1c84000       4       4       4 rw---   [ anon ]
00007ffd8c16e000     132       8       8 rw---   [ stack ]
00007ffd8c1db000      12       0       0 r----   [ anon ]
00007ffd8c1de000       4       4       0 r-x--   [ anon ]
ffffffffff600000       4       0       0 --x--   [ anon ]
---------------- ------- ------- -------
total kB          821480   56736   55412
```

Man sieht das mehr Speicher reserviert wird als übergeben. Dies ist nicht verwunderlich, da nur der Array den übergebenen Speicher nimmt. Dazu muss man aber natürlich noch den Rest des Programms oben drauf zählen. An den Screenshots sieht man, wenn man 500mb oder 800mb dem Array zuweist, kommt natürlich am Ende mehr belegter Speicher raus.

Dirty pages werden auf die Festplatte geschrieben, wenn der Arbeitsspeicher zu voll wird. Im ersten Beispiel sind es sogar 125MB, im zweiten lediglich 55MB obwohl mehr Speicher allokiert wurde.

RSS ist die Resident set size und ist die Größe in KB, die tatsächlich auf dem Arbeitsspeicher liegt. Bei der linken Ausführung sind es 126MB und bei der rechten knapp 57MB.