



## Übungen zur Speicherverwaltung

Laden Sie das Programmarchiv zur aktuellen Übung mit

```
wget http://swf.hgesser.de/b1-ss2016/prakt/swf-bs1-ss2016-uebung05.tgz
```

von der Webseite herunter, entpacken Sie es mit `tar xzf swf-bs1-ss2016-uebung05.tgz` und wechseln Sie in den neuen Ordner `ue05/`.

- Aufgabe 2: **in Heimarbeit**;  
Lösung ab 07.07. online
- Aufgabe 1: während der Vorlesung

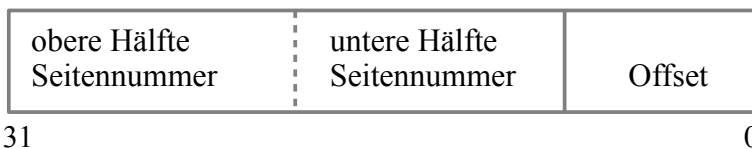
### 1. Speicherverwaltung: Paging (Theorie)

Ein Betriebssystem mit virtueller Speicherverwaltung arbeite mit

- 32 Bit langen virtuellen Adressen,
- einer Seitengröße von 1 KByte,
- 2-stufigem Paging, wobei die äußere und die inneren Seitentabellen gleich groß sind,
- Seitentableneinträgen der Länge 4 Byte.

a) Wie sieht das Format einer virtuellen Adresse aus, d. h., welche der 32 Bits der Adresse haben welche Bedeutung?

(Überlegen Sie zunächst, wie viele Bits für den Offset verwendet werden – daraus ergibt sich die Anzahl der Bits für die kompletten Seitennummern, durch Halbieren dann die Anzahl der Bits von unterer/oberer Hälfte der Seitennummer.)



b) Wie viele innere Seitentabellen gibt es? Wie groß sind die äußere bzw. die inneren Seitentabellen?

### 2. Speicherverwaltung Linux/C (Praxis)

Sehen Sie zur Vorbereitung die Folien F51–F59 an.

Betrachten Sie das auf Seite 2 abgedruckte Programm, das Sie auch als `memory-test.c` im Ordner `ue05/` finden. Es reserviert vier Speicherbereiche (für `global`, `on_stack`, `malloced1` und `malloced2`) und zeigt deren zugewiesene Speicheradressen an; danach ruft es das externe Programm `pmap` auf, das die Speicherzuteilung für den laufenden Prozess ausgibt.

a) Kompilieren Sie das Programm und lassen Sie es laufen.

b) Versuchen Sie, eine Zuordnung zwischen den vier ausgegebenen (Start-) Adressen sowie den Speicherbereichen in der `pmap`-Ausgabe zu finden.



- c) Die Größe der Speicherbereiche wird an zwei Stellen (`global`, `on_stack`) über `sizeof()` abgefragt, aber für `malloced1` und `malloced2` ist das nicht der Fall – und es ist auch nicht möglich. Woran liegt das und welche Ausgabe würde `sizeof(malloced1)` bzw. `sizeof(malloced2)` zurückgeben?
- d) Wenn Sie das Programm mehrfach ausführen, sehen Sie jedesmal leicht abweichende Adressen für mehrere der Speicherbereiche. Woran könnte das liegen? (Tipp: Googlen Sie die Abkürzung ASLR.)

```
// memory-test.c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define MALLOC1_SIZE 1024*1024*16
#define MALLOC2_SIZE 1024

char global[1024*1024*32];    // 32 MB global

void subroutine (char *arg1, char *arg2) {
    char on_stack[1024*1024*2]; // 2 MB on stack

    printf ("global:    %010p (%5d K)\n", global, sizeof(global)/1024);
    printf ("on_stack:  %010p (%5d K)\n", on_stack,
        sizeof(on_stack)/1024);
    printf ("malloced1: %010p (%5d K)\n", arg1, MALLOC1_SIZE/1024);
    printf ("malloced2: %010p (%5d K)\n", arg2, MALLOC2_SIZE/1024);

    char cmd[30];
    sprintf ((char*)cmd, "pmap %d | grep -v /lib", getpid());
    system (cmd);
}

int main () {
    char *malloced1 = malloc (MALLOC1_SIZE); // 16 MB malloc
    char *malloced2 = malloc (MALLOC2_SIZE); // 1 KB malloc
    subroutine (malloced1, malloced2);
}
```