#uni/semester3/Betriebssysteme/chapter18/homework

1. Before doing any translations, let's use the simulator to study how linear page tables change size given different parameters. Compute the size of linear page tables as different parameters change. Some suggested inputs are below; by using the -v flag, you can see how many page-table entries are filled. First, to understand how linear page table size changes as the address space grows, run with these flags:
   -P 1k -a 1m -p 512m -v -n 0
   -P 1k -a 2m -p 512m -v -n 0
   -P 1k -a 4m -p 512m -v -n 0

   ➜ Larger address space, means more pages, which means more translations, therefore more entries in the page table.

   Then, to understand how linear page table size changes as page size grows:

   -P 1k -a 1m -p 512m -v -n 0
   -P 2k -a 1m -p 512m -v -n 0
   -P 4k -a 1m -p 512m -v -n 0

   ➜ Larger page sizes, means less pages in one address space, which means less translations and therefore less entries in the page table.

   Before running any of these, try to think about the expected trends. How should page-table size change as the address space grows? As the page size grows? Why not use big pages in general?
   ➜ Big pages results in wasted memory, because it won't fill everything at the start. Why do big pages, when you can do smaller, but mark those that are not needed as invalid.

2. Now let's do some translations. Start with some small examples, and change the number of pages that are allocated to the address space with the -u flag. For example:
   a) -P 1k -a 16k -p 32k -v -u 0
   b) -P 1k -a 16k -p 32k -v -u 25
   c) -P 1k -a 16k -p 32k -v -u 50
   d) -P 1k -a 16k -p 32k -v -u 75
   e) -P 1k -a 16k -p 32k -v -u 100

➜ How to calculate everything, see the read me. For the calculations see paper in Ordner

What happens as you increase the percentage of pages that are allocated in each address space?
➜ Page table grows?

3. Now let's try some different random seeds, and some different (and sometimes quite crazy) address-space parameters, for variety:

   -P 8 -a 32 -p 1024 -v -s 1
   ➜ too small page size and address space
   -P 8k -a 32k -p 1m -v -s 2
   ➜ I would say, these parameters are still too small
   -P 1m -a 256m -p 512m -v -s 3
   ➜ These seem okay, but for modern systems still too small. Especially the size of physical memory

   Which of these parameter combinations are unrealistic? Why?

4. Use the program to try out some other problems. Can you find the limits of where the program doesn't work anymore? For example, what happens if the address-space size is bigger than physical memory?
   ➜ I get an error if address space size is larger than physical memory
   ➜ Error when page size is not a power of 2
   ➜ Error when address space is not a power of 2
   ➜ IndexError when page size is larger than address space