CSE 451: Section 6 Handout
2/13/2020

Page Faults, Stack, and Heap

Page Faults:
A trap 14 defines a page fault, this means that the memory address accessed is not mapped or the access protection is violated (write to read-only page).

Stack:
Can the kernel cause a page fault that was meant for stack growth?

Write some C user level code that causes a page fault for stack growth.
You may assume:
- page size is 4096 Bytes
- the user stack starts with 1 page mapped

Heap:
Look at the code below. It runs as expected and prints out 1 2 3 (each on a new line).
Why doesn't the bolded line cause the program to crash, even though we only allocated one byte with sbrk? (Note how the bolded line writes past that boundary).

```
#include <stdio.h>
#include <unistd.h>

int main(int argc, char** argv) {
  char *ptr;
  ptr = (char *)sbrk(1);
  *(ptr + 1) = 1;
  *(ptr + 2) = 2;
  *(ptr + 3) = 3;
  printf("%d\n", *(ptr + 1));
  printf("%d\n", *(ptr + 2));
  printf("%d\n", *(ptr + 3));
  return 0;
}
```