**INLAB. 7 ( 2 )**

\* free\_and\_sbrk.c

Test if free(3) actually lowers the program break.

Usage: free\_and\_sbrk num-allocs block-size [step [min [max]]]\*/

#define MAX\_ALLOCS 1000000

#include <stdio.h> /\* Standard I/O functions \*/

#include <stdlib.h> /\* Prototypes of commonly used library functions,plus EXIT\_SUCCESS and EXIT\_FAILURE constants \*/

#include <unistd.h> /\* Prototypes for many system calls \*/

#include <errno.h> /\* Declares errno and defines error constants \*/

#include <string.h> /\* Commonly used string-handling functions \*/

int main(int argc, char \*argv[])

{

char \*ptr[MAX\_ALLOCS];

int freeStep, freeMin, freeMax, blockSize, numAllocs,j;

printf("\n");

if (argc < 3 || strcmp(argv[1], "--help") == 0)

{

printf("%s num-allocs block-size [step [min [max]]]\n" argv[0]);

exit(5);

}

numAllocs = strtol(argv[1], NULL, 10);

if (numAllocs > MAX\_ALLOCS)

{

printf("num-allocs > %d\n", MAX\_ALLOCS);

exit(5);

}

blockSize = strtol(argv[2], NULL, 10);

freeStep = (argc > 3) ? strtol(argv[3], NULL, 10): 1;

freeMin = (argc > 4) ? strtol(argv[4], NULL, 10) : 1;

freeMax = (argc > 5) ? strtol(argv[5], NULL, 10) : numAllocs;

if (freeMax > numAllocs)

{

printf("free-max > num-allocs\n");

exit(5);

}

printf("Initial program break: %10p\n", sbrk(0));

printf("Allocating %d\*%d bytes\n", numAllocs, blockSize);

for (j = 0; j < numAllocs; j++)

{

ptr[j] = malloc(blockSize);

if (ptr[j] == NULL)

{

perror("malloc");

exit(5);

}

}

printf("Program break is now: %10p\n", sbrk(0));

printf("Freeing blocks from %d to %d in steps of %d\n",freeMin, freeMax, freeStep);

for (j = freeMin -1;

j < freeMax;

j += freeStep)free(ptr[j]);

printf("After free(), program break is: %10p\n", sbrk(0));

exit(10);

}

**OUTPUT:**

osdlab:~$ cc free\_and\_sbrk.c

osdlab:~$ ./a.out 1000 10240 2

The output shows that after these blocks have been freed, the program break is left unchanged from the level it reached when all memory blocks were allocated:

Initial program break: 0x1e1f000Allocating 1000\*10240 bytes

Program break is now: 0x27e7000

Freeing blocks from 1 to 1000 in steps of 2After free(),

program break is: 0x27e7000

The following command line specifies that all but the last of the allocated blocks should be freed. Again, the program break remains at its “high-water mark.”

osdlab:~$ ./a.out 1000 10240 1 1 999

Initial program break:

0xed7000Allocating 1000\*10240 bytes

Program break is now: 0x189f000

Freeing blocks from 1 to 999 in steps of 1

After free(), program break is: 0x189f000

If, however, we free a complete set of blocks at the top end of the heap, we see that the program break decreases from its peak value, indicating that free() has used sbrk() to lower the program break. Here, we free the last 500 blocks of allocated memory:

osdlab:~$./a.out 1000 10240 1 500 1000

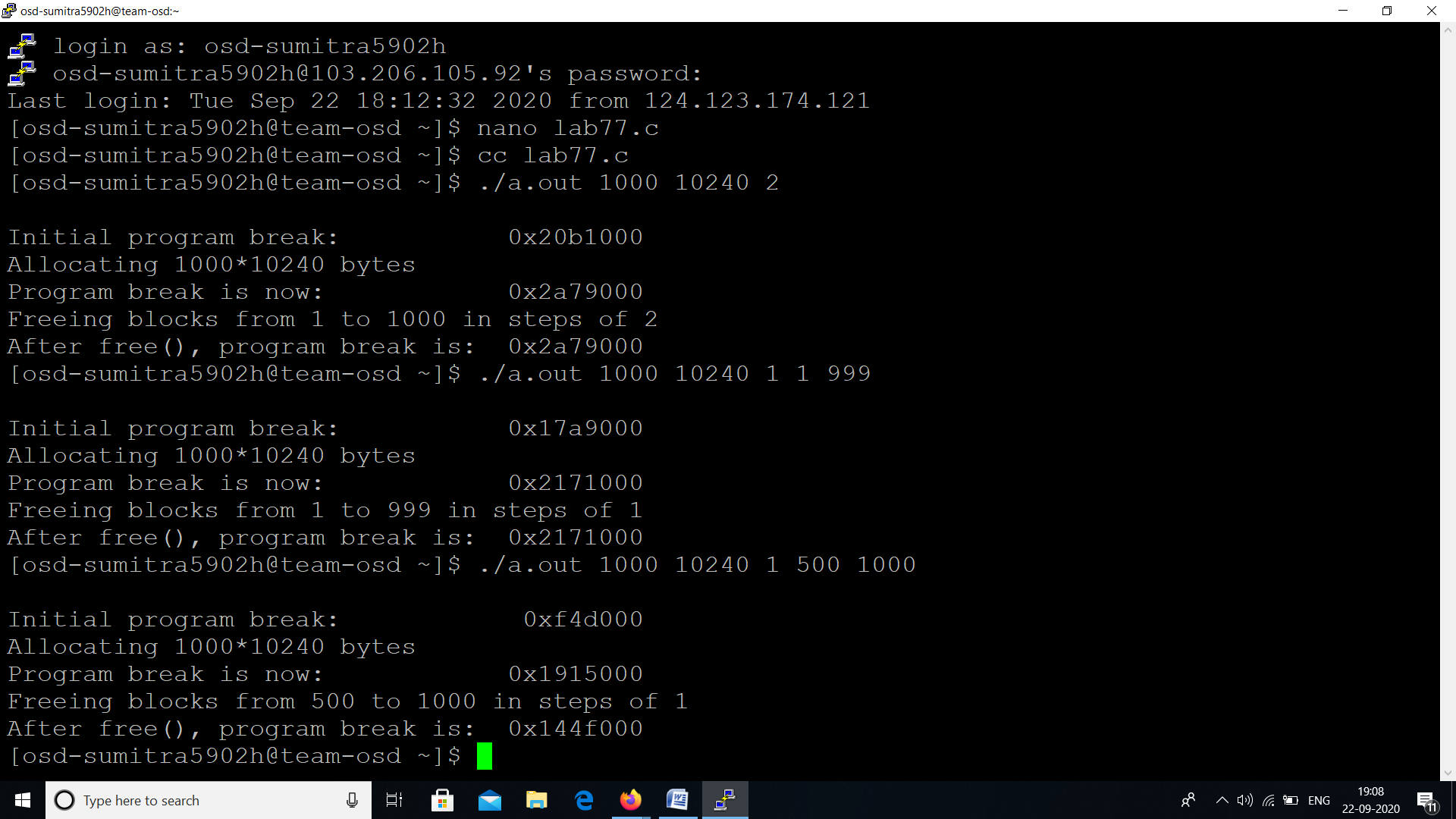
Initial program break: 0x211c000

Allocating 1000\*10240

bytesProgram break is now: 0x2ae4000

Freeing blocks from 500 to 1000 in steps of 1

After free(), program break is: 0x25fd000

****