PRACTICA 5pids

Created	@January 15, 2021 10:31 AM
Date	@November 9, 2020 → November 22, 2020
Tags	
Tema	

Tema i què es fa

Gestión de memoria. Reserva de espacios y regiones de memoria para procesos.

Sessió i codis

▼ mem1_v2.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define DEFAULT_ITERATIONS  3
#define REGION_SIZE   4096

char globalArray[REGION_SIZE];

int
main (int argc, char *argv[])
{
  char buff[80];
  int niterations = DEFAULT_ITERATIONS;
  int i;
  char localArray[REGION_SIZE];
  char *p;

  if (argc > 2)
    {
      sprintf (buff, "Usage: mem1 [numMallocs]\n");
      write(2,buff,strlen(buff));
      exit (1);
    }
```

```
  if (argc == 2)
    {
      niterations = atoi (argv[1]);
    }

  sprintf (buff, "Addresses:\n");
  write(1,buff,strlen(buff));
  sprintf (buff, "\tglobalArray: %p\n", globalArray);
  write(1,buff,strlen(buff));
  sprintf (buff, "\tlocalArray: %p\n", localArray);
  write(1,buff,strlen(buff));
  sprintf (buff, "\tp: %p\n", &p);
  write(1,buff,strlen(buff));


  for (i = 0; i < niterations; i++)
    {
      p = malloc (REGION_SIZE);
      sprintf (buff, "\tp: %p, region %d: %p\n", &p, i, p);
      write(1,buff,strlen(buff));
      free(p);
    }

  while (1);

}
```

▼ mem1_v3.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define DEFAULT_ITERATIONS  3
#define REGION_SIZE   4096

char globalArray[REGION_SIZE];

int
main (int argc, char *argv[])
{
  char buff[80];
  int niterations = DEFAULT_ITERATIONS;
  int i;
  char localArray[REGION_SIZE];
  char *p;

  if (argc > 2)
    {
      sprintf (buff, "Usage: mem1 [numMallocs]\n");
      write(2,buff,strlen(buff));
      exit (1);
    }

  if (argc == 2)
    {
      niterations = atoi (argv[1]);
    }

  sprintf (buff, "Addresses:\n");
  write(1,buff,strlen(buff));
  sprintf (buff, "\tglobalArray: %p\n", globalArray);
  write(1,buff,strlen(buff));
  sprintf (buff, "\tlocalArray: %p\n", localArray);
  write(1,buff,strlen(buff));
  sprintf (buff, "\tp: %p\n", &p);
  write(1,buff,strlen(buff));
```

```
  for (i = 0; i < niterations; i++)
    {
      p = sbrk(REGION_SIZE);
      sprintf (buff, "\tp: %p, region %d: %p\n", &p, i, p);
      write(1,buff,strlen(buff));
    }

  while (1);

}
```

▼ mem2_v2.c

```
#include <stdio.h>
#include <signal.h>
#include <stdlib.h>
#include <string.h>

#define REGION_SIZE   4096

int *p;

void func(int signal)
{
  char buff[256];
        sprintf( buff, "\tProgram code -- p address: %p, p value: %p, p heap value: %p\n", &p, p, sbrk(0));
  write(1, buff, strlen(buff));
  exit(0);
}

int main(int argc, char *argv[])
{
  struct sigaction sa;
  sa.sa_handler = &func;
  sa.sa_flags = SA_RESTART;
  sigfillset(&sa.sa_mask);
  sigaction(SIGSEGV, &sa, NULL);

  int i = 0;
  char buff[256];

  sprintf( buff, "Addresses:\n");
  write(1, buff, strlen(buff));
  sprintf( buff, "\tp: %p\n", &p);
  write(1, buff, strlen(buff));

  p = malloc(sizeof(int));

  if (p == NULL) {
    sprintf(buff, "ERROR en el malloc\n");
    write(2,buff,strlen(buff));
  }

  while (1) {
    *p = i;
    sprintf( buff, "\tProgram code -- p address: %p, p value: %p, *p: %d\n",
          &p, p, *p);
    write(1, buff, strlen(buff));
    p++;
    i++;
  }

}
```

Manual

❓ malloc - Valida una región de memoria

❓ free - Libera una región de memoria

❓ sbrk - Modifica el tamaño de la región de datos