PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU
FACULTAD DE CIENCIAS E INGENIERIA

Laboratorio Nro 4
(2021 - 1)

1) El esquema básico de un programa es el siguiente:

```
/**********  mapstack.c *********/
#include <stdio.h>

int main(int narg, char *argv[], char *env[])
{    int a;

     return 0;
}
```

El objetivo es obtener un pequeño mapa de la pila. Lo que nos interesa saber es el orden en que se encuentra las siguientes estructuras: a) las variables automáticas `a` y `narg`, b) el arreglo `argv[]`, c) los argumentos de la línea de ordenes, d) el arreglo `env[]` y e) las variables del entorno. El orden debe ser proporcionado desde las posiciones de memoria más bajas, hasta las más altas.

Como primer paso imprima las direcciones de las estructuras anteriormente mencionadas. Una vez que sabe cuál es el orden, agregue impresiones (*printf*) apropiadas para que dicho orden sea evidente cuando se ejecute el programa. También se debe demostrar que la dirección más baja y la más alta de las estructuras están dentro del segmento de la pila (6 puntos).

Una vez establecido el orden demuestre, también mediante impresiones (*printf*) apropiadas, si cada elemento está una a continuación de otro. Por ejemplo, si después de b) siguiera e) la pregunta sería: ¿está b) a continuación de e) o hay un gap? Conteste esta pregunta por cada estructura (6 puntos).

Para que su programa imprima el inicio y fin del segmento de pila de su proceso, puede incorporar este programa:

```c
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

int main(int narg, char *argv[])
{  FILE * file;
   int fd,pidof;
   char orden[50];
   char salida[100];

   pidof=getpid();
   sprintf(orden,"cat /proc/%d/maps | grep stack ",pidof);
   file=popen(orden,"r");
   fgets(salida,sizeof(salida),file);
   printf("%s",salida);
   return 0;
}
```

Nota: no se olvide de ejecutar el programa con argumentos en la línea de ordenes, por ejemplo:

```
./mapstack INF239 Sistemas Operativos
```

2) (8 puntos) En esta pregunta usted trabajará en dos terminales. En la primera, editará y compilará el programa, en la segunda lo ejecutará como superusuario.

En la terminal donde lo ejecutará, escriba lo siguiente:

```
sudo -i
```

Proporcione la contraseña e inmediatamente ubíquese en el directorio donde se encuentra sus programas fuentes. En la línea de ordenes escriba:

```
export TEXTO1=0x120
export TEXTO2=0x130
```

En la primera terminal complete el siguiente programa:

```c
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int narg, char *argv[], char *env[])
{  int pidof;
   char path[20];

   if(pidof=fork()){ /*Padre*/
       putenv("TEXTO1=Sistemas");
       putenv("TEXTO2=Operativos");
   }
   else {  /*Hijo*/
        int d;

   }
   wait(NULL);
   return 0;
}
```

El proceso hijo lee de */proc/[pid]/mem* del proceso padre e imprime cada una de las variables de entorno. El programa es correcto si logra imprimir `TEXTO1=Sistemas` y `TEXTO2=Operativos`.

No puede usar otros programas.

Lima, 11 de junio de 2021.

Prof: Alejandro T. Bello Ruiz.