Instituto Tecnológico de Costa Rica

Escuela de Computación

Curso: Sistemas Operativos

Tarea 8

Un *breakpoint* es un momento en la ejecución de un programa que se encuentra almacenado y al cuál se puede regresar (*rollback*) utilizando dicha información almacenada. De esta forma una tarea podría volver atrás a algún momento anterior de su ejecución.

La librería *libtask* administra directamente los datos de la pila de cada tarea, esto hace posible el copiar los datos a un archivo y luego recuperarlos. Se ha adicionado una nueva función a *libtask* llamada *taskstack()* que retorna la dirección a la pila de la tarea que se está ejecutando. El siguiente ejemplo muestra un programa que calcula la secuencia de fibonacci, la forma de invocarlo es, por ejemplo:

```
breakpoint 30
```

Cuando el contador del ciclo llega a 10, el programa hace una copia de la pila en *buffer* y posteriormente escribe estos datos a un archivo.

```
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <task.h>
#include <fd.h>

#define SAVE 0
#define LOAD 1

enum { STACK = 32768 };

char buffer[STACK] = "";
int mode=LOAD;

void savebuffer(char *filename, char *buffer, int size) {
  FILE *f1;
  if ((f1 = fopen(filename, "w")) == -1)  taskexit(-1);
  fdnoblock(f1);
  if (fdwrite(f1, buffer, size) != size)
    printf("cp: write error on file %s\n", filename);
```

```
  fclose(f1);
}

void fibonacci(void *arg) {
  int n, first = 0, second = 1, next, c;
  char *stk;
  n = (int)arg;
  for (c = 0; c < n; c++) {
    if (c == 10) {
      stk = taskstack();
      mode=SAVE;
      memcpy(buffer,&stk,STACK);
      if (mode==SAVE)
        savebuffer("break1", buffer, STACK);
    }
    if (c <= 1)
      next = c;
    else {
      next = first + second;
      first = second;
      second = next;
    }
    printf("%d\n", next);
  }
}

void taskmain(int argc, char **argv) {
  if (argc == 0) {
    printf("Usage: breakpoint n\n");
    taskexit(-1);
  }
  taskcreate(fibonacci, atoi(argv[1]), STACK);
}
```

Ejercicios

1. Modifique el programa anterior de forma que, al llegar el ciclo principal a 20, regrese al estado cuando el contador estaba en 10 (*rollback*). Muestre ejemplos de la correcta ejecución de su programa.

2. Intente *resucitar* una tarea, es decir, volver a un estado anterior de una tarea que ya terminó. Tome en cuenta que para realizar esto debe crear una instancia de la misma tarea que ya finalizó, pero no debe permitir que esta tarea calcule nada sino que debe sobrescribir su pila utilizando la que se encuentra almacenada en el archivo. Muestre ejemplos de la correcta ejecución de su programa.