CC3301 Programación de Software de Sistemas

Control 2 – Semestre Primavera 2013 – Prof.: Luis Mateu

Pregunta 1

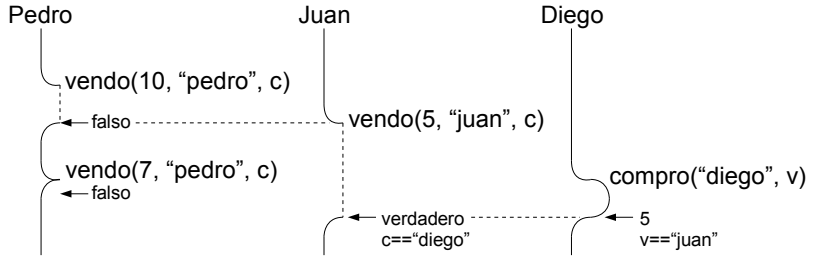
Las acciones de la empresa ACME se transan en una bolsa de comercio, cuyos operadores son representados mediante threads. Para comprar o vender una acción los operadores usan las funciones *compro* y *vendo*. Sus encabezados se indican en el cuadro siguiente, junto a ejemplos de uso.

```
int vendo(int precio, char *vendedor, char *comprador);
int compro(char *comprador, char *vendedor);
```

```
char *nom= miNombre();
int precio= miPrecio();
char comprador[100];
if (vendo(precio, nom,
          comprador)) {
  printf("vendi a %s\n",
         comprador);
}
```

```
char *nom= miNombre();
char vendedor[100];
sleep(tiempoAleatorio());
int precio= compro(nom, vendedor);
if (precio>0) {
  printf("compre a %s en %d\n",
         vendedor, precio);
}
```

A la izquierda se muestra el código usado por los múltiples vendedores de acciones y a la derecha el código de los múltiples compradores. La función *compro* transa una sola acción con el vendedor más barato de ese momento, entregando el precio pagado y copia el nombre del vendedor en el *2do parámetro*. Si en ese momento no hay ningún vendedor el precio será 0 y *compro* retorna de inmediato. La función *vendo* ofrece una acción al precio indicado y espera hasta que (i) aparezca un comprador, en cuyo caso retorna verdadero y copia el nombre del comprador en el 3er parámetro, o (ii) aparezca (o ya hay) un vendedor con un precio menor, retornando falso en tal caso. El siguiente diagrama muestra un ejemplo de ejecución.



Pedro llama a *vendo*, que retorna falso cuando Juan llama a *vendo* con un precio menor. La segunda llamada de Pedro fracasa de inmediato porque su precio todavía es mayor al de Juan. Juan sí tiene éxito cuando Diego llama a *compro* y por lo tanto la llamada a *vendo* de Juan retorna verdadero, copiando el nombre "diego" en el parámetro *comprador*. Por su parte *compro* retorna 5 (el precio) y copia "juan" en el parámetro *vendedor*.

Programe las funciones *vendo* y *compro*. Para la sincronización debe usar un monitor (es decir, un mutex y una condición).

Pregunta 2

Parte a.- (3 puntos) Considere las siguientes declaraciones:

```
typedef int (*Comparador)(void *obj1, void *obj2);
typedef struct nodo {
  void *obj;
  struct nodo *prox;
} Nodo;
void insertar(Nodo **cabeza, Comparador comp, void *obj);
```

Comparador es el tipo de las funciones que comparan 2 objetos, retornando -1, 0 o 1 según *obj1* es respectivamente menor, igual o mayor que *obj2*. El tipo *Nodo* corresponde a un elemento de una lista enlazada de objetos genéricos. La función *insertar* agrega el objeto *obj* a una lista ordenada ascendentemente de acuerdo al orden inducido por el comparador *comp*. Por ejemplo en el siguiente código:

```
  Nodo tercero= { "tal", NULL };
  Nodo segundo= { "que", &tercero };
  Nodo primero= { "hola", &segundo };
  Nodo *cabeza= &primero;
  insertar(&cabeza, (Comparador)strcmp, "susto");
```

La lista resultante debe ser "hola", "que", "susto", "tal".

Programe la función *insertar*.

Parte b.- (1,5 puntos) Use la parte a.- para escribir un programa que lee las líneas de no más de 80 caracteres de la entrada estándar y las entrega en *orden alfabético descendente* en la salida estándar. Ejemplo de uso:

```
% cat datos.txt
que
hola
tal
```

```
% ordenar < datos.txt
tal
que
hola
%
```

Recuerde que la función `fgets(s, 81, stdin)` lee una línea de hasta 80 caracteres de la entrada estándar. El parámetro *s* debe apuntar a un espacio en memoria de 81 bytes en donde quedará la línea leída como un string de C (en donde el '\n' fue reemplazado por '\0'). La función retorna NULL si se llegó al final de la entrada estándar.

Parte c.- (1,5 puntos) Use la parte a.- para programar una función que ordene ascendentemente un arreglo de enteros. Ejemplo de uso:

```
int i;
int arreglo[]= {6, -2, 4, 3, 10 };
ordenar_arreglo(arreglo, 5);
for (i= 0; i<5; i++) printf("%d ", arreglo[i]);
/* muestra -2, 3, 4, 6, 10 */
```