EXAMEN SISTEMAS OPERATIVOS AVANZADOS TEORIA

VIERNES 15 DICIMEBRE 2017

NOMBRE:

1. ¿Qué resultado obtendremos por pantalla en los siguientes tres trozos de código?

int x[10]; printf("%d\n", sizeof(x));	int x[10]; printf("%d\n", sizeof(&x));	int x[10]; printf("%d\n", sizeof(*x));

2. Corregir el siguiente código

```
char *src = "hello";
char *dst;
strcpy(dst, src);
```

3. ¿Es el siguiente código correcto? Corregir el problema/s si no lo fuera.

```
Segment = (VirtualAddress & SEG_MASK) >> SEG_SHIFT
Offset    =   VirtualAddress & OFFSET_MASK
if (Offset >= Bounds[Segment])
PhysAddr = Base[Segment] + Offset
if ( PhysAddr >= Bounds[Segment] )
    RaiseException (PROTECTION_FAULT)
Else:
    Register = AccessMemory(PhysAddr)
```

4. Completar que va en "????????" para gestinar una free list de forma embebida

```
typedef struct __node_t {
  int size;
  struct __node_t *next;
} nodet_t;
```

```
node_t *head = mmap(NULL, 4096, PROT_READ|PROT_WRITE,\
                MAP_ANON|MAP_PRIVATE, -1, 0);
head->size = 4096 - ???????;
head->next = NULL;
```

5. Imaginar un caso en el que la PTE contiene solo contiene el número de marco y un bit de validez (no incluye ni bit de presencia, bit de dirty, etc…). Este bit lo tiene situado en el menos significativo de la entrada. Suponer que tenemos un espacio de direcciones virtual de 128 bytes con páginas de 32 bytes. El contenido de la tabla de páginas es: 0xFE, 0x0F, 0xFE, 0x09. ¿Qué dirección física se accederá en cada una de las siguientes direcciones virtuales?
   - 0x065
   - 0x00c
   - 0x026
   - 0x058
6. Suponer una tabla de páginas en dos niveles. Suponer un espacio de direcciones virtual de 15-bits con páginas de 32bytes. El formato de PDE y PTE es el mismo: un bit de validez seguido de 7 bits para referenciar el marco de página. El PDBR vale 73 en decimal. A continuación, se muestra un volcado parcial de la memoria:

```
pa 6: 0a 1c 01 14 0b 1a 19 0a 0a 1a 0c 14 02 0c 1c 0c 15 04 0e 13 17 11 08 05 08 07 04 13 0f 1d 0f 1e
pa 73: a2 d2 97 96 d9 7f 87 b4 b7 f2 f4 82 bf 7f be 93 e8 9d 99 9e f1 7f 7f b0 d8 da eb b1 81 c3 c2 f6
pa 114:7f 7f 7f 7f 82 7f 7f 7f 7f 7f 7f 7f 99 7f 7f 7f 7f 7f 7f 86 7f 7f 7f 7f 7f 7f 8f 7f 7f 7f 7f 7f
```

Se pretende acceder a la dirección virtual 0x1787. ¿Qué traducción se obtiene?

7. ¿Qué implicaciones tiene el siguiente cambio en la implementación de un *lock* basado en swap atómico (*x86*)? ¿Sigue funcionando?

```
while(1){
  while (lock > 0) ;
  if (xchg(&lock, 1) == 0)
    return;
}
```

8. Indicar y corregir el/los problema/s que tiene la siguiente implementación de la inserción de una lista thread-safe

```
typedef struct _node_t {
  Int key;
  __node_t *next;
}
mutex_t m = PTHREAD_MUTEX_INITIALIZED
note_t *head = NULL;

int List_insert(int key)
{
  mutex_lock(&m);
  node_t *n = malloc(sizeof(__node_t);
  assert  (n != NULL);
  n->key = key;
  n->next = head;
  head = n;
  mutex_unlock(&m);
  return 0;
}
```

9. Proponer los cambios necesarios para que el siguiente código sea *thread-safe* sin emplear exclusión mutua (suponiendo que estamos empleando un procesador x86).

```
Int _counter;
void initCounter()
{
  _counter = 0;
}
void incrementCounter()
{
  _counter++;
}
int getCounter()
{
  return _counter;
}
```

10. Aunque la siguiente implementación del ***Zemaphores*** tiene un *bug*, en determinadas circunstancias puede funcionar "bien" ¿Cuándo?

```
typedef struct __Zem_t {
  int value;
  pthread_cond_t cond;
  pthread_mutex_t lock;
} Zem_t;

// only one thread can call this
void Zem_init(Zem_t *s, int value) {
  s->value = value;
  Cond_init(&s->cond);
  Mutex_init(&s->lock);
}
```

```
void Zem_wait(Zem_t *s) {
  Mutex_lock(&s->lock);
  if (s->value <= 0)
    Cond_wait(&s->cond, &s->lock);
  s->value--;
  Mutex_unlock(&s->lock);
}
void Zem_post(Zem_t *s) {
  Mutex_lock(&s->lock);
  s->value++;
  Cond_signal(&s->cond);
  Mutex_unlock(&s->lock);
}
```

EXAMEN SISTEMAS OPERATIVOS AVANZADOS PRACTICAS

VIERNES 15 DICIMEBRE 2017

NOMBRE:

Examen disponible en https://gitlab.com/AOSUC/examen2. En el directorio “xv6/tests” de cada Práctica hay dos ficheros “Q1.c” y “Q2.c” asociados a dos tests que no pasan correctamente (cada uno de ellos debido a errores en una sola función). Escribir el código corregido de cada función (¡en el espacio reservado!).

LAB 3

Q1.c 2.5 (2.5 Puntos)

Q2.c 2.5 (2.5 Puntos)

LAB 4

Q1.c 2.5 (2.5 Puntos)

Q2.c 2.5 (2.5 Puntos)