Examen Mejoramiento Programación de Sistemas

2do Término 2017 – 2018

Nombre: _____

Pregunta 1 (25 puntos)

Considere que tiene una máquina de 32-bits, Little Endian. El arreglo **a** empieza en la dirección 1000, el arreglo **c** en la dirección 8000 y el buffer **z** en la dirección 12000 (ver código en el reverso). Muestre lo siguiente: La salida de las sentencias en printfs en (1) y (2), el contenido del buffer (3) después del ***for*** (byte por byte), empezando desde la dirección 12000.

Parte 1

8	+2
24	+2

Parte 2

-3	+1
4	+1
FD FF FF FF	+4

Parte 3

12000 a 12003 --> 8 0 0 0	+1.5
12004 a 12007 --> f 0 0 0	+1.5
12008 a 12011 --> 9 0 0 0	+1.5
12012 a 12015 --> e 0 0 0	+1.5
12016 a 12019 --> a 0 0 0	+1.5
12020 a 12023 --> d 0 0 0	+1.5
12024 a 12027 --> b 0 0 0	+1.5
12028 a 12031 --> c 0 0 0	+1.5
12032 a 12035 --> c 0 0 0	+1.5
12036 a 12039 --> b 0 0 0	+1.5

```c
int main(void){

	int *z = (int *) malloc(10*sizeof(int));
	int a[5] = {8,9,10,11,12};
	int c[5] = {11,12,13,14,15};
	int *b = &a[3];
	int *d = (c+2);

	//1
	printf("%lu\n", (long)&a[4] - (long)&a[2]);
	printf("%d\n", *(d+1) + 10);

	//2
	int v = *(b+1) - *(d+2);
	printf("%d\n",v);

	unsigned char *p = (unsigned char *)&v;

	printf("size of int: %lu\n", sizeof(int));

	printf("%x\n", *p);
	printf("%x\n", *(p+1));
	printf("%x\n", *(p+2));
	printf("%x\n", *(p+3));
```

Pregunta 2 (25 puntos)

A partir del siguiente código (el código continúa en el reverso), haga lo siguiente:

1. Indique cuántos hilos son ejecutados y qué función es invocada cuando se inicia cada hilo.
2. Escriba la salida del programa.

```
#define WORKERS 3
#define CONTROLLERS 2

void *worker_thread(void *vargp);
void *controller_thread(void *vargp);

volatile int mine[WORKERS*CONTROLLERS+1] = { 0 };
sem_t s;
pthread_t w_tid[WORKERS];

int main()
{
    int c, i, *ptr;
    pthread_t c_tid;
    Sem_init(&s, 0, 1);
        for (c = 0; c < CONTROLLERS; c++) {
            for (i = 0; i < WORKERS; i++) {
                ptr = Malloc(sizeof(int));
                *ptr = i+(WORKERS*c);
                Pthread_create(&w_tid[i], NULL, worker_thread, ptr);
            }
            ptr = Malloc(sizeof(int));
            *ptr = c;
            Pthread_create(&c_tid, NULL, controller_thread, ptr);
            Pthread_join(c_tid, NULL);
        }
    for(i = 0; i < c*WORKERS; i++)
        printf("%d\n", mine[i]);
    printf("\n");
    exit(0);
}

void *worker_thread(void *vargp)
{
    int myid = *((int *)vargp);
    Free(vargp);
    P(&s);
    mine[WORKERS*CONTROLLERS]++;
    mine[myid] = ++myid * myid;
```

```
void *worker_thread(void *vargp)
{
    int myid = *((int *)vargp);
    Free(vargp);
    P(&s);
    mine[WORKERS*CONTROLLERS]++;
    mine[myid] = ++myid * myid;
    V(&s);
    return NULL;
}

void *controller_thread(void *vargp)
{
    int i, c_tid = *((int *)vargp)+1;
    Free(vargp);
    for(i = 0; i < WORKERS; i++)
        Pthread_join(w_tid[i], NULL);
    for(i = 0; i < c_tid*WORKERS; i++){
        P(&s);
        mine[WORKERS*CONTROLLERS]+=mine[i];
        V(&s);
    }
    printf("Controller %d summary: %d", c_tid,
                        mine[WORKERS*CONTROLLERS]);
    printf("\n");
    return NULL;
}
```

El programa genera 9 hilos:

1 hilo principal que trabaja con la función “main”	+1
2 hilos que invocan la función controller_thread	+1
6 hilos que invocan la función worker_thread	+1

Salida:

`Controller 1 summary: 17`	+5
`Controller 2 summary: 111`	+5
`1`	+2
`4`	+2
`9`	+2
`16`	+2
`25`	+2
`36`	+2

Pregunta 3 (25 puntos)

Implemente la siguiente función:

```
int readline(int fd, void *usrbuf, int maxlen)
```

La función ***readline*** debe de leer el descriptor de archivos *fd* usando **read()** hasta encontrar un salto de línea, encontrar EOF o hasta leer *maxlen* bytes. Todos los datos leídos deben de ser grabados correctamente en *usrbuf* (asuma que este buffer tiene ya su memoria reservada). La función debe de retornar el número de bytes leídos, incluyendo el salto de línea, o -1 si hubo algún error.

```
int readline(int fd, void *usrbuf, int maxlen)
{
	int n, rc;
	char c;
	char *bufp = usrbuf;
	//Intenta leer todo hasta maxlen: +5
	for(n = 1; n < maxlen; n++) {
		//Usa read correctamente, 1 byte por lectura: +5
	rc = read(fd, &c, 1);
		//Valida retorno del read = 1 : +5
		if (rc == 1) { //Todo bien, leyó un byte
			*bufp++ = c; //guarda en el buffer
			if(c == '\n') { //Si en salto de línea, termina
				n++;
				break;
			}
		//Valida retorno del read = 0 : +5
		}else if (rc == 0) { //Leyó EOF
			//No había ningún caracter leido previo, retorna 0
			if (n == 1)
				return 0;
			else //Había caracteres previos
				break;
		}else //Hubo algún error
			//Valida retorno del read = -1 : +5
			return -1;
	}
```

```
        *bufp = 0;
        return n - 1;
}
```

Pregunta 4 (25 puntos)

¿Cuáles son las 3 maneras de construir programas concurrentes, provistas por sistemas operativos modernos? Indique las características de cada modelo.

- **Procesos:**	**+2**
- Cada flujo lógico es un proceso. - Contextos separados por procesos y manejados por Kernel. - Espacios de memoria separados.	**+7**
- **I/O Multiplexing:**	**+2**
- Varios flujos lógicos dentro de proceso único (único contexto) - Mismo espacio de Memoria.	**+5**
- **Hilos:**	**+2**
- Flujos lógicos ejecutándose dentro del mismo proceso. - Contextos por hilos - Mismo espacio de memoria.	**+7**