Guía de ejercicios 1
Ejercicios sobre SO y procesos

1. ¿Cuál es la diferencia entre *System Call* y *Procedure Call*?

2. ¿Cuál es la diferencia entre un *System Call* y un cambio de contexto?

3. ¿Cuál es la diferencia entre el *Kernel Stack* y el *User Stack*?

4. ¿Qué se almacena en el *Heap*?

5. ¿Qué bandera en EFLAGS para la arquitectura x86 apoya el proceso de cambio de contexto?

6. Explique las diferencias entre un SO basado en procesos y un SO que se ejecuta en el contexto de procesos usuarios.

7. Para el siguiente programa dibuje el espacio de direcciones del proceso en el momento que se está ejecutando la función `void f(char*)`. Indique el contenido de cada segmento.

```
int i=0;
int *A;
main() {
  int j;
  A = malloc(10*sizeof(int));
  for (j=0; j < 10; j++)
    f(A);
}

void f(char *p) {
  int z;
  for (z=0; z < 10; z++)
    printf(" %d\n", A[z]);
}
```

8. Cree un esquema donde muestre el proceso que ocurre a nivel de proceso usuario y kernel cuando ocurre un syscall.

9. Para el siguiente código:

```
main()
{
  char *p;
  p = malloc(1);

  printf("%p\n", (void*) main);
  printf("%p\n", (void*) p);
  printf("%p\n", (void*) &p);
}
```

Asuma que %p muestra la dirección (en hexadecimal) del objeto (variable). Considerando la estructura de un proceso vista en clases ¿En qué segmentos caen las direcciones mostradas por pantalla? Explique.

10. ¿Qué mecanismos por parte del sistema operativo se gatillan cuando corremos el siguiente programa ya compilado a través de la bash? Describa el proceso.

```
int fibo(n){
        int i, res=0, suc=1, sucsuc=1;
        if(n<=1) return 1;
        while(n>1){
                res= suc + sucsuc;
                sucsuc=suc;
                suc=res;
                n--;
        }
        return res;
}

main(){
      fibo(n));
}
```