Exámen Final – 22 de diciembre de 2021

Sistemas Operativos – OSTEP

Virtualización

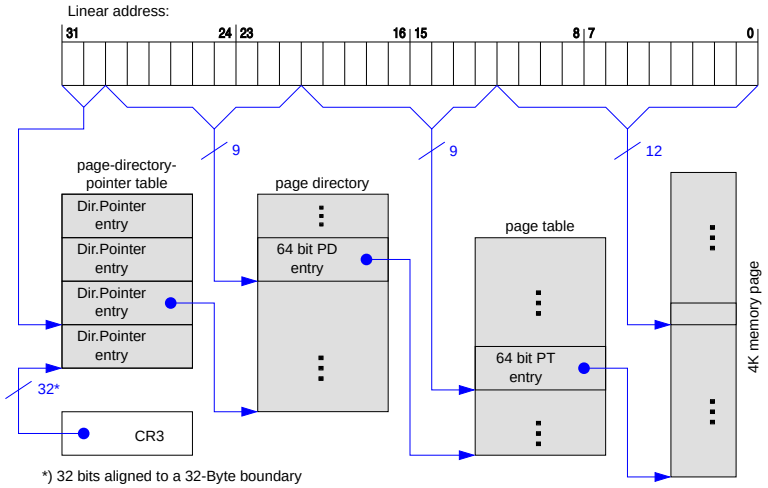
Ejercicio 1. Explique detalladamente como funcionan estos programas indicando cuantos procesos se crean, que hacen los padres, etc.
Suponga que en ambos casos se invocan con

```
$ ./a.out ./a.out ./a.out ./a.out ./a.out ./a.out ./a.out
```

```
int main(int argc, char ** argv) {
  char buf[L] = {'\0'};
  if (argc<=1)
    return 0;
  int rc = fork();
  if (rc<0)
    return -1;
  if (0==rc) {
    close(2);
    open(argv[0], 0);
    read(2, buf, L);
    write(1, buf, L);
  } else {
    argv[argc-1] = NULL;
    execvp(argv[0], argv);
  }
}
```

```
int main(int argc, char ** argv) {
  if (argc<=1)
    return 0;
  int rc = fork();
  if (rc<0)
    return -1;
  else if (0==rc)
    main(argc-2, argv)
  else {
    argv[argc-1] = NULL;
    argv++;
    execvp(argv[0], argv);
  }
}
```

Ejercicio 2. Para el sistema de paginado i386 con PAE de $(2, 9, 9, 12)$ que transforma de 32 bits virtual a 36 bits físicos. Los marcos físicos ahora pasan de 20 bits (5 dígitos hexa) a 24 bits (6 dígitos hexa).



Sabiendo que `CR3=0x20000` y que el contenido de los marcos físicos son los siguientes:

```
0x000000                0x010000                0x020000
----------------        ----------------        ----------------
0x000: 0x010000 (P)     0x000: 0x000000 (P)     0x000: 0x000000 (P)
0x001: 0x010000 (P)     0x001: 0x000001 (P)     0x001: 0x000000 (P)
...                     ...                     0x002: 0x020000 (P)
...                     ...                     0x003: 0x000000 (P)
0x1FE: 0x010000 (P)     0x1FE: 0x0001FE (P)
0x1FF: 0x000000 (P)     0x1FF: 0x0001FF (P)
```

(a) Pasar de virtual a física: `0x00000EEF`, `0x00001FEE`, `0x00200EEF`, `0x00201FEE`.

(b) Pasar de virtual a física `0x80000CFF`, `0x8001F000`

(c) Pasar de física a todas las virtuales `0x000000FFF`, `0x020000FFF`, `0x030000FFF`

Ejercicio 3.

(a) Muestre un escenario de ejecución que termine en `a={0,1,0,1,0,1 ...}` para la Figura 1, suponga siempre **atomicidad es línea-a-línea**.

(b) Agregue sincronización con semáforos para que el valor de salida sea siempre el del punto anterior. Puede colocar condicionales (`if`) sobre el valor de las variables `i` y `j` para hacer wait/post de los semáforos. No puede tocar los incrementos de las variables y las asignaciones al arreglo.

Pre: $0 <$`N` $\wedge$ `i,j`$=0,0 \wedge$ $(\forall k : 0 \leq k < N :$ `a`$[k] = 2)$	

`1 P0: while (i<N) {`
`2  a[i] = 0;`
`3  ++i;`
`}`
	`a P1: while (j<N) {`
	`b  a[j] = 1;`
	`c  ++j;`
	`}`

Figura 1: Multiprograma

Ejercicio 4. Considere los procesos `P0` y `P1` a continuación, donde las sentencias **no son** atómicas.

Pre: `n`$= 0$	

`P0 : while (n<100) {`
`n = n*2;`
`}`
	`P1 : while (n<100) {`
	`n = n+1;`
	`}`

(a) ¿Qué valores posibles puede tomar `n` al terminar el multiprograma?. Explique.

(b) Sincronizar con semáforos para que se comporte como la versión atómica dada en el práctico que da valores de `n` entre 100 y 200.

Ejercicio 5. Usted forma parte de un equipo al que le fue encargado el diseño de un sistema de archivos *UNIX-like* y uno de sus compañeros propuso la idea de de aprovechar los 75 bytes que sobran en los i-nodos (Cada i-nodo ocupa 181 bytes y se decidió "alinearlo" a 256 bytes para que entraran 2 por bloque de disco) almacenando allí los archivos que no superen esta capacidad[1].

(a) **Discuta** porque si o porque no esta **decisión de diseño** es acertada, en cuanto a eficiencia, velocidad y simpleza. Soporte sus argumentos con las mediciones[2] de la Figura 2.

(b) ¿Tiene sentido que estos 75 bytes **siempre se utilicen** sin importar la longitud del archivo? Discuta.

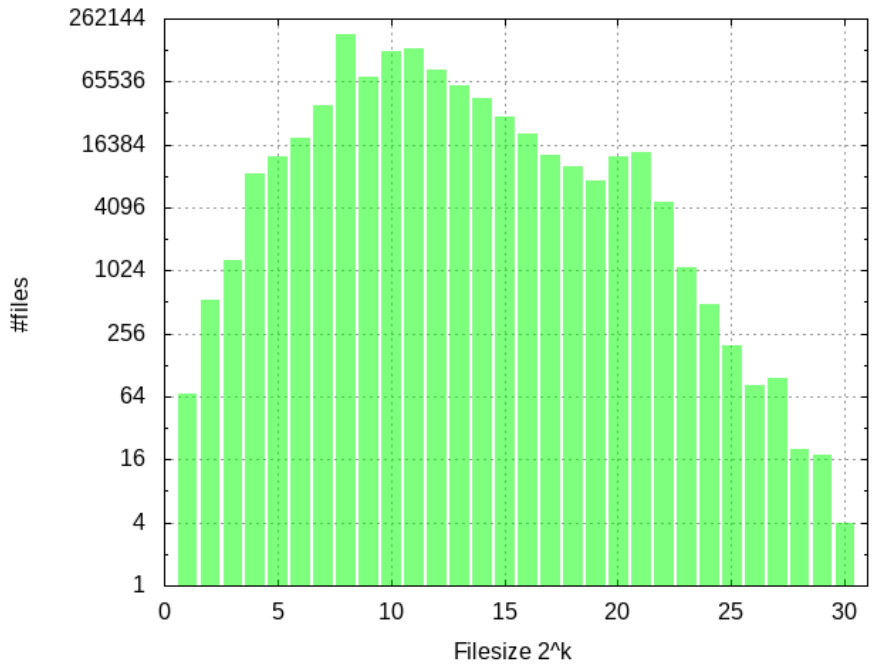


Figura 2:

Ejercicio 6. En un sistema de archivos de tipo UNIX, tenemos los bloques de disco dispuestos dentro del *i-nodo* con 12 bloques directos, 1 bloque indirecto y 1 bloque doble indirecto. Cada bloque es de 4 KiB.

(a) Calcule la capacidad máxima del *block pool* para números de bloque de 16, 24 y 32 bits.

(b) Calcule la capacidad máxima de un archivo para números de bloque de 16, 24 y 32 bits.

(c) Realice un análisis de que longitud de número de bloque conviene.

[1] Esto ya existe y se conoce como ***immediate files***, tanto en MinixFS como en NTFS.

[2] La figura muestra la cantidad de archivos que ocupan determinado tamaño en todo mi disco duro al día 20141214. Por ejemplo tengo aproximadamente un poco más de 65536 archivos de tamaño entre $2^9 = 0{,}5\ KiB$ y $2^{10} = 1\ KiB$