Sistemas Operativos. Práctica 1.

Pablo Cuesta Sierra y Álvaro Zamanillo Sáez

Semana 1

Ejercicio 1

a) Para buscar las funciones relacionadas con hilos, buscamos todas aquellas que contengan "pthread" usando la opción -k en el comando man. El comando a usar es:

```
$ man -k pthread
```

b) Consultar en el manual en qué sección se encuentran las llamadas a sistema y buscar infromación sobre write:

`man man`: con este comando averiguamos que la sección relacionada a *system calls* es la 2. Por lo tanto usamos el comando `man 2 write`.

Ejercico 2

a) El comando empleado [soper1] es: **grep -C 1 -w molino don \ quijote.txt >> aventuras.txt**.

Usamos el comando grep para buscar las apariciones de 'molino'. Como queremos que sea la palabra molino y no el grama molino añadimos el parámetro `-w`. Además para incluir las lineas donde aparece, usamos -C 1 indicando así que deseamos que se incluya una línea de contexto. Finalmente redireccionamos usando >> en vez de > para que se añada al contenido ya existente en vez de borrarlo.
b) El *pipeline* es el siguiente **ls | wc -l**

La salida de ls es una lista (fichero) con los ficheros del actual directorio. Esta lista la usamos como input del comando wc, que acompañado de la opción -l, cuenta las lineas de dicha lista.
c) En este caso el *pipeline* es **cat lista\ de\ la\ compra\ Pepe.txt lista\ de\ la\ compra\ Elena.txt 2>/dev/null | uniq > numcompra.txt**

Si no existe algún archivo de los dos, se produce un error al concatenar, ahí es donde usamos el redireccionamiento a /dev/null en caso de error. Si no ha sucedido ningún error al concatenar, usamos la salida de cat como input del

comando uniq (usando pipeline). Finalmente, la salida del comando uniq la redireccionamos a "numcompra.txt"

Ejercicio 3

a) Si intentamos abrir un fichero inexistente recibimos el mensanje ""No such file or directory". El código errno asociado es 2.
b) En el caso de intentar abrir el fichero /etc/shadow el mensaje de error es "Permissoon denied" que corresponde al valor de errno 13.
c) Justo después de la instrucción fopen() se debería guardar el valor de errno en otra variable ya que la llamada a perror() podría modificar la variable global errno.

pf=fopen(args[1],"r"); x=errno; perror();

Ejercicio 4

a) Durante los 10 segundos de espera el proceso asociado al programa está en estado ”R” (runnable), es decir se está ejecutando (intercalándose con otros).
b) En este caso el proceso está en estado "S" (interruptible sleep) o lo que es lo mismo, esperando a un suceso (el fin de la espera marcada por sleep()).

Ejercicio 5

`cosas`

Search 8. [fig:ser8]

```
bool findKey(const char * book_id, const char *indexName, int * nodeIDOrDataOffset){
    FILE *pf=NULL;
    int pos, comp;
    Node node;
    bool found = false;
    char b_id[5], search[5];

    if(!indexName)
        return false;

    if((pf=fopen(indexName,"rb"))==NULL)
        return false;

    memcpy(search, book_id, PK_SIZE);
    search[PK_SIZE] = '\0';
    /*read the root's position*/
    fread(&pos, sizeof(int), 1, pf);

    if(pos==-1){/*empty*/
```

```
        *nodeIDOrDataOffset=-1;
    }

    while(pos != -1 && found == false){
        /*find the node's offset in the file*/
        fseek(pf, INDEX_HEADER_SIZE+pos*sizeof(Node), SEEK_SET);
        /*read the node*/
        fread(&node, sizeof(Node), 1, pf);
        memcpy(b_id, node.book_id, PK_SIZE);
        b_id[PK_SIZE] = '\0';
        /*compare the primary key*/
        comp=strcmp(search, b_id);

        if(comp < 0){
            if(node.left==-1)/*not found*/
                (*nodeIDOrDataOffset) = pos;
            pos = node.left;
        }
        else if(comp > 0){
            if(node.right==-1)/*not found*/
                (*nodeIDOrDataOffset) = pos;
            pos = node.right;
        }
        else{/*found*/
            (*nodeIDOrDataOffset) = node.offset;
            found = true;
        }
    }
    fclose(pf);
    return found;
}
```