**SISTEMES OPERATIUS II: Pràctica 1**

**1. A l’enunciat de la pràctica es comenta que la funció fgets és més segura que fscanf per llegir text pla. Podeu comentar per què? Quines avantatges aporta la funció fgets a nivell de seguretat?**

La funció fgets es mes segura a la hora de llegir textos plans ja que llegeix linea per línia els caràcters. A la vegada també se li pot dir el nombre màxim de caràcters que vols que llegeixi. En comparació amb el fscanf que segueix un cert codi de format que tu li demanes.

Per temes de seguretat la avantatge que ens aporta el fgets es que va emmagatzemant el text que llegeix per cada línia, en comparació amb el fscanf que no emmagatzema les dades que va llegint.

**2. Com s’ha adaptat el codi de la manipulació de l’arbre perquè la clau d’indexació sigui una cadena. Indiqueu de forma clara quines funcions s’han adaptat. Recordeu que, tal com s’indica a la secció 2.1, cada node ha d’emmagatzemar la paraula amb la mida mínima necessària (i.e. no es pot suposar que la paraula té una mida màxima de, per exemple, 100 caràcters). Podeu incloeu codi per facilitar la descripció de les modificacions realitzades.**

Tant al mètode compare\_key1\_less\_than\_key2 com compare\_key1\_equal\_to\_key2 de la classe red-black-tree hem canviat els paràmetres de la funció. Inicialment aquests venien amb un “int” ja que comparava nombres. Nosaltres per poder dur a terme la funcionalitat de la practica vam haver de canviar-los per char a partir d’un punter. També hem canviat la funció internament, ja que abans es comparaven els dos números i nosaltres hem afegit una funció per que compari el string sense sensibilitat a l’hora de ser minúscula o majúscula.

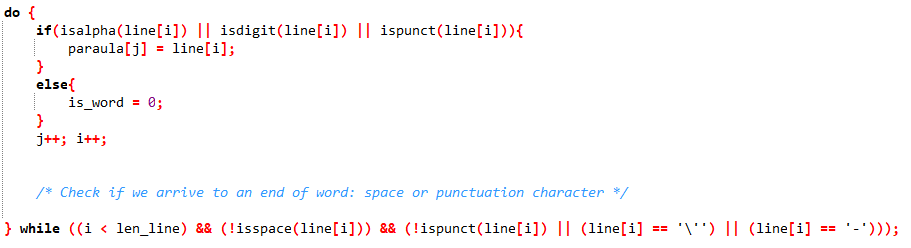
Finalment per la part de la classe red-black-tree a la funció “find\_node” hem canviat el paràmetre de key de tipus int a un char.

**3. Quin és el problema que té el codi d’extracció de paraules de la secció 2.2? Comenteu breument com ho heu arreglat.**

El problema que tenim al codi d’extracció de paraules es que extreu les paraules independentment dels signes de puntuació que trobem a la paraula(ens referim als guions, apòstrofs). Es a dir, de la paraula taper-light, al trobar-se amb el guió la separa en dues paraules.

Aquest problema esdevé en la classe extraccio\_paraules.c, ja que de la línia sencera agafarà una paraula, anirà comprovant caràcter a caràcter si es un caràcter alfanumèric, en el cas que no ho sigui, es a dir hi hagi un espai o sigui un signe de puntuació començarà a comptar una altra paraula.

Per la qual cosa tal com s’ha demanat a la practica per evitar que corroeixin aquests casos a l’hora de comprovar si la paraula ha arribat al final, creem una nova condició que comprovi si el caràcter que li toca imprimir es un apòstrof(‘) o un guio(-).



**4. En cas que incloeu alguna cosa excepcional (diferent) de la demanada a l’enunciat, podeu incloure també l’explicació de la funcionalitat implementada.**

No s’ha afegit cap altre cosa excepcional de les demanades a l’enunciat.

**5. Quines són les 10 paraules que apareixen més cops als fitxers de texts proporcionats? Comenteu quantes vegades apareixen cadascuna d’aquestes 10 primeres paraules i indiqueu clarament com heu obtingut el resultat. Sou capaços d’obtenir el resultat ajudant-vos de les instruccions de la línia de comandes? És a dir, podeu obtenir el resultat sense haver**

**de programar-ho tot en C?**