NOM		DATA	**25 / 05 / 2015**	QUALIFICACIÓ
ÀREA/MATÈRIA	**DAW - M03 - UF3**	CURS	**2014 - 2015**	

Escolliu 10 preguntes de l'examen. Feu captures de pantalla per a documentar l'examen.

L'objectiu de les primeres 11 preguntes és desenvolupar un generador d'arxius XML d'aquest estil :

```
<?xml version="1.0" encoding="utf-8"?>
<arxius>
        <tipus t="ascii">
                <ext e=".c">
                        <fitxer f="ex01.c"></fitxer>
                        <fitxer f="ex02.c"></fitxer>
                        <fitxer f="ex03.c"></fitxer>
                        <fitxer f="ex04.c"></fitxer>
                        <fitxer f="ex05.c"></fitxer>
                        <fitxer f="ex06.c"></fitxer>
                </ext>
                <ext e=".txt">
                        <fitxer f="tlb.txt"></fitxer>
                </ext>
        </tipus>
        <tipus t="binari">
                <ext e=".pdf">
                        <fitxer f="arxiu.pdf"></fitxer>
                </ext>
        </tipus>
</arxius>
```

1)***(1 punt)*** Compileu ex02.c que mostra pel terminal el llistat d'arxius presents al directori d'execució.

2)***(1 punt)*** Desenvolupeu ex03.c que mostra pel terminal el llistat de les extensions dels arxius presents al directori d'execució.

```
void vMostraExtensions(FILE *f){
        char szLinia[N_MIDA_NOM],*pStr;

        while (fgets(szLinia,N_MIDA_NOM,f) != NULL){
                pStr = szLinia;
                while(!(*pStr == '\n' || *pStr == '.')  && *pStr)
                        pStr++;
                printf("%s",pStr);
        }
}

A la funció principal comenteu vLecturaLlistaDir(f) i afegiu vMostraExtensions(f) :

//      vLecturaLlistaDir(f);
        vMostraExtensions(f);
        fclose(f);
```

3)***(1 punt)*** Desenvolupeu ex04.c que conté la funció `int nComptaArxiusPuntC(FILE *f)` que compta els arxius amb extensió .c presents al directori d'execució.

```
Podeu aprofitar el codi de void vMostraExtensions(FILE *f) i afegir :

                //printf("%s",pStr);
                if(! strncmp(pStr,".c",2))
                        nCompta++;
        }
        return nCompta;


A la funció principal comenteu vLecturaLlistaDir(f) i vMostraExtensions(f) per a provar-ho :

//      vLecturaLlistaDir(f);
//      vMostraExtensions(f);
        printf("Arxius amb extensió .c : %d\n",nComptaArxiusPuntC(f));
        fclose(f);
```

4)***(1 punt)*** Desenvolupeu ex05.c que conté la funció `int nComptaArxiusExtensio(FILE *f,char* szExt)` que compta els arxius amb l'extensió passada com a segon argument presents al directori d'execució.

```
Podeu aprofitar el codi de int nComptaArxiusPuntC(FILE *f) canviant :

                if(! strncmp(pStr,szExt,strlen(szExt)))
                        nCompta++;
        }
        return nCompta;


A la funció principal comenteu les proves prèvies :

//      vLecturaLlistaDir(f);
//      vMostraExtensions(f);
//      printf("Arxius amb extensió .c : %d\n",nComptaArxiusPuntC(f));
        printf("Arxius amb extensió .c : %d\n",nComptaArxiusExtensio(f,".c"));
        rewind(f);
        printf("Arxius amb extensió .txt : %d\n",nComptaArxiusExtensio(f,".txt"));
        rewind(f);
        printf("Arxius amb extensió .pdf : %d\n",nComptaArxiusExtensio(f,".pdf"));
        fclose(f);

        return 0;

Recordeu que rewind retorna l'apuntador del fitxer al seu inici
```

5)***(1 punt)*** Desenvolupeu ex06.c que conté les funcions `void vEscriuCapXml(char *sz)` i `void vEscriuPeuXml(char *sz)` que mostren pel terminal la capçalera i el peu d'un arxiu XML.

```
void vEscriuCapXml(char *sz){
        sprintf(sz,"<?xml version=\"1.0\" encoding=\"utf-8\"?>\n<arxius>\n");
}

void vEscriuPeuXml(char *sz){
        char szAux[N_GROS];

        sprintf(szAux,"</arxius>\n");
        strcat(sz,szAux);
}

A la funció principal comenteu les proves prèvies i afegiu:

        vEscriuCapXml(szXml);
        vEscriuPeuXml(szXml);
        printf("Contingut arxiu XML :\n%s\n",szXml);
        fclose(f);

        return 0;

Definiu la cadena szXml[N_MOLT_GROS] a l'inici del main ( #define N_MOLT_GROS 99999 )
```

6)***(1 punt)*** Desenvolupeu ex07.c que conté la funció `void vEscriuTipus(char *sz,char *szTipus)` que mostren pel terminal els tipus de l'arxiu XML.

```
A la funció principal :

        vEscriuCapXml(szXml);
        vEscriuTipus(szXml,"ascii");
        vEscriuTipus(szXml,"binari");
        vEscriuPeuXml(szXml);
        printf("Contingut arxiu XML :\n%s\n",szXml);
        fclose(f);

Execució :

$ ./ex07
Contingut arxiu XML :
<?xml version="1.0" encoding="utf-8"?>
<arxius>
        <tipus t="ascii">
        </tipus>
        <tipus t="binari">
        </tipus>
</arxius>
```

7)***(1 punt)*** Desenvolupeu ex08.c que conté la funció `void vEscriuExtensio(char *sz,char *szTipus)` que es cridada per `void vEscriuTipus(char *sz,char *szTipus)` mostrant pel terminal les extensions pels diferents tipus d'arxiu :

```
Execució :

$ ./ex08
Contingut arxiu XML :
<?xml version="1.0" encoding="utf-8"?>
<arxius>
        <tipus t="ascii">
                <ext e=".c">
                </ext>
                <ext e=".txt">
                </ext>
        </tipus>
        <tipus t="binari">
                <ext e=".pdf">
                </ext>
        </tipus>
</arxius>

Pista :

        sprintf(szAux,"\t<tipus t=\"%s\">\n",szTipus);
        if(!strcmp(szTipus,"ascii")){
                vEscriuExtensio(szAux,".c");
                vEscriuExtensio(szAux,".txt");
        }
        if(!strcmp(szTipus,"binari")){
                vEscriuExtensio(szAux,".pdf");
        }
        strcat(szAux,"\t</tipus>\n");
        strcat(sz,szAux);
```

8)***(1 punt)*** Desenvolupeu ex09.c que conté la funció `void vEscriuNomArxius(FILE *f,char *sz,char *szTipus)` mostrant pel terminal els fitxers del directori d'execució. Per a fer això s'ha d'afegir un argument FILE *f a aquestes funcions cridades des de la funció principal :

```
        vEscriuCapXml(szXml);
        vEscriuTipus(f,szXml,"ascii");
        vEscriuTipus(f,szXml,"binari");
        vEscriuPeuXml(szXml);
        printf("Contingut arxiu XML :\n%s\n",szXml);
        fclose(f);

        return 0;

La funció vEscriuNomArxius és cridada per:

void vEscriuExtensio(FILE *f,char *sz,char *szTipus){
        char szAux[N_GROS];

        sprintf(szAux,"\t\t<ext e=\"%s\">\n",szTipus);
        vEscriuNomArxius(f,szAux,szTipus);
        strcat(szAux,"\t\t</ext>\n");
        strcat(sz,szAux);
}

Part inicial de l'execució :

$ ./ex09
Contingut arxiu XML :
<?xml version="1.0" encoding="utf-8"?>
<arxius>
        <tipus t="ascii">
                <ext e=".c">
                        <fitxer f="02_substituirText_03.c
"></fitxer>
```

9)***(1 punt)*** Desenvolupeu ex10.c que conté la funció `char* szSenseSaltLinia(char *sz)` mostrant pel terminal l'arxiu XML definitiu:

```
     while (fgets(szLinia,N_MIDA_NOM,f) != NULL){
             pStr = szLinia;
             while(!(*pStr == '\n' || *pStr == '.')  && *pStr)
                     pStr++;
             if(! strncmp(pStr,szTipus,strlen(szTipus))){
                     sprintf(szAux,"\t\t\t<fitxer f=\"%s\"></fitxer>\n",szSenseSaltLinia(szLinia));
                     strcat(sz,szAux);
             }
     }
```

```
$ ./ex10
Contingut arxiu XML :
<?xml version="1.0" encoding="utf-8"?>
<arxius>
     <tipus t="ascii">
             <ext e=".c">
                     <fitxer f="ex01.c"></fitxer>
                     <fitxer f="ex02.c"></fitxer>
                     <fitxer f="ex03.c"></fitxer>
                     <fitxer f="ex04.c"></fitxer>
                     <fitxer f="ex05.c"></fitxer>
                     <fitxer f="ex06.c"></fitxer>
                     <fitxer f="ex07.c"></fitxer>
                     <fitxer f="ex08.c"></fitxer>
                     <fitxer f="ex09.c"></fitxer>
                     <fitxer f="ex10.c"></fitxer>
             </ext>
             <ext e=".txt">
                     <fitxer f="arxius.txt"></fitxer>
                     <fitxer f="desredundat.txt"></fitxer>
                     <fitxer f="tirantLoBlanc.txt"></fitxer>
                     <fitxer f="tlb.txt"></fitxer>
             </ext>
     </tipus>
     <tipus t="binari">
             <ext e=".pdf">
                     <fitxer f="20120522ex.pdf"></fitxer>
             </ext>
     </tipus>
</arxius>
```

10)***(1 punt)*** Desenvolupeu ex11.c que escriu l'arxiu XML amb el nom ex.xml

11)***(1 punt)*** Desenvolupeu ex12main.c, ex12.c i ex12.h esmicolant ex11.c en un projecte de tres arxius.

12)***(1 punt)*** Implementeu la funció de còpia d'un fitxer a un altre. El prototipus de la funció ha de ser :

```
void copia (char *f_origen, char *f_desti)
```

f_origen i **f_desti** són els noms dels fitxers. Són els noms reals dels fitxers, tal i com els tenim desats al nostre ordinador, per exemple origen.txt i desti.txt

13)***(1 punt)*** Implementeu una acció similar a l'anterior que realitzi una còpia del fitxer amb redundància. Això ho farem copiant cadascun dels caràcters del fitxer origen n vegades al fitxer destí.

Per exemple, si el fitxer origen conté:
Fitxer origen: `People have the power`

Per una redundància de 3 el resultat seria:
Fitxer destí: `PPPeeeoooppplllleee hhhaaavvveee ttthhheee pppooowweeerrr`

La capçalera de l'acció serà:

```
void copia_redundancia (char *f_origen, char *f_desti, int redun)
```

f_origen: nom real del fitxer origen de les dades
f_desti: nom real del fitxer destí
redun: nombre de vegades que cal repetir cada caràcter. En l'exemple seria 3

14)***(1 punt)*** Implementeu una acció per recuperar un fitxer que s'ha copiat amb redundància. Suposarem que no hi ha hagut cap error en la creació del fitxer i que el fitxer s'ha redundat correctament.

La capçalera de la funció seria:

```
void recupera_redundancia (char *f_origen, char *f_desti, int redun)
```

f_origen: nom real del fitxer que conté la redundància
f_desti: nom real del fitxer on recuperarem la informació sense redundància
redun: nombre de vegades que està repetit cada caràcter

Fes servir la funció fgets. Recorda que aquesta funció es crida com a l'exercici 2.

15)***(1 punt)*** Programa una funció que donat un fitxer creat amb redundància, detecti quina és la seva redundància. En l'exemple anterior hauria de retornar un 3

Per fer-ho cal recórrer els primers caràcters del fitxer i comptar tots els que són iguals.

La capçalera de la funció seria:

```
int detecta_redun (char *f_origen)
```

f_origen: nom real del fitxer origen de les dades
return: retorna el nombre de vegades que es repeteixen els caràcters, per tant la redundància

16)***(1 punt)*** Reprogrameu l'acció recupera_redundància per tal de detectar automàticament la redundància. En aquest cas la capçalera seria:

```
void recupera_redundancia2 (char *f_origen, char *f_desti)
```

17)***(1 punt)*** Implementeu una funció que ens permeti saber si tots els caràcters d'una cadena són iguals

```
int tots_iguals (char *cadena, int longi)
```

cadena: és la cadena de caràcters que volem comprovar si són iguals
longi: és la longitud de la cadena
return: 1 si tots són iguals, 0 si no ho són

18)***(1 punt)*** Reprogrameu l'acció recupera_redundancia3 per tal de detectar possibles errors. En aquest cas, cada vegada que llegim els n caràcters redundants utilitzarem la funció tots_iguals per saber si tots els caràcters són iguals.

En el cas que no sigui iguals vol dir que hi ha un error. En aquest cas copiarem el caràcter #
En el cas que siguin iguals tot ha anat bé. Copiarem el caràcters que toca.

Exemple:

Aquesta cadena amb redundància:

`PPPe_eooop-pllleee hhhaaavvveee ttthhheee pppooowwweeerrr`

Hauria de retornar:

`P#o#le have the power` (al no trobar els tres caràcters consecutius posa el símbol #)

La capçalera serà:

```
void recupera_redundancia3 (char *f_origen, char *f_desti)
```

19)***(1 punt)***Implementeu un programa principal amb dues opcions:

Opció 1: La primera opció permetrà copiar un fitxer amb redundància. Cal demanar a l'usuari el fitxer origen, fitxer destí i redundància que li vol aplicar.

Opció 2: Permetrà recuperar un fitxer creat amb redundància. Cal demanar a l'usuari el fitxer origen i el fitxer destí. Origen és el fitxer amb redundància i destí és el fitxer recuperat.

Molta sort a tots !!!!