Une image contenant texte, carte de visite

Description générée automatiquement



Il apparaît opportun de commencer ce rapport par des remerciements à ceux qui ont eu la gentillesse de le rendre très profitable. Au terme de ce travail, Nous tenons à exprimer nos vifs remerciements à :

* Mme Touhami Ouazzani Khadija
* Mr Lazrek Mohamed

Nous vous remercions d'avoir partagé avec nous votre passion pour l'enseignement. Nous avons grandement apprécié votre soutien, votre implication et votre expérience tout au long de l'année.

Sommaire

1. Description du Projet
2. Travail réalisé
3. Code Source
4. Exécution du programme
5. Conclusion

Description du Projet :

L'objectif du projet Structure de données avancées est de mettre en œuvre un système de fichiers hiérarchique simple avec stockage uniquement dans la mémoire principale. Un système de fichiers hiérarchique organise les ressources selon une structure arborescente et identifie de manière unique chaque ressource via le chemin qui la connecte à la racine. Les ressources contenues dans un système de fichiers hiérarchique peuvent être des fichiers ou des répertoires. Les deux ont un nom, représenté par une chaîne de caractères. Pour les premiers, il est seulement possible de les insérer comme feuilles de l'arbre, tandis que les seconds peuvent apparaître à la fois comme feuilles et comme nœuds intermédiaires. La racine de l'arbre est classiquement un répertoire, appelé répertoire racine. Seuls les nœuds de fichiers peuvent contenir des données, représentées comme une séquence d'octets, tandis que les répertoires n'ont pas de données associées. Tous les nœuds de l'arborescence peuvent contenir des métadonnées, mais aux besoins de ce projet, seuls les répertoires les contiennent. Les métadonnées du répertoire sont les noms de ses descendants directs.

Le programme qui implémente le système de fichiers recevra un journal des actions à effectuer à partir de l'entrée standard et imprimera le résultat de la même chose sur la sortie standard. Le programme doit être implémenté en C standard. Le fonctionnement du programme nécessite qu'il lise une ligne du journal des actions, et exécute l'action correspondante sur la représentation interne du système de fichiers qu'il gère, et écrive le résultat sur la sortie standard avant de passer à l'action suivante (l'exécution des actions est complètement séquentielle).

Travail réalisé :

Les chemins du système de fichiers sont représentés avec la syntaxe UNIX habituelle : un chemin est donc la séquence de noms de ressources qui du répertoire racine atteignent la ressource identifiée par le chemin. Les noms sont séparés par le caractère séparateur de chemin /). Par exemple, considérons le système de fichiers suivant :

/

Fich3

Fich2

Fich1

Rep1

Rep2

Fich0

Le chemin qui identifie la ressource de fichier 0 est / Fich0, celui qui identifie le fichier 3 est / Rep2 / Fich3.

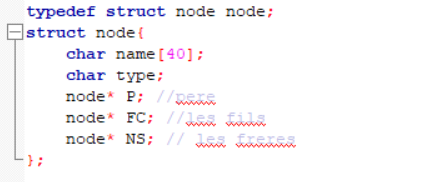
La structure d’arbre n’aire s’applique.

Le programme reçoit l'une des commandes suivantes pour chaque ligne du fichier journal donné en entrée, où il indique un chemin générique et une chaîne alphanumérique avec un maximum de 255 caractères.

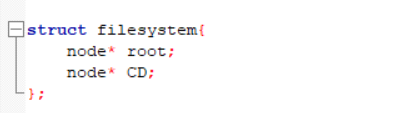
* + Créer-f : la commande crée un fichier vide ou sans données associées dans le système de fichiers. Imprimer le résultat "fichier créé avec succès" si le fichier a été créé régulièrement, "Impossible de créer le fichier" si la création du fichier n'a pas réussi (par exemple, si vous essayez de créer un fichier dans un répertoire inexistant ou si le nom du fichier existe déjà).
  + Créer-rep : créez un répertoire vide dans le système de fichiers. Imprimez le résultat " répertoire créé avec succès " si le répertoire a été créé régulièrement, " Impossible de créer le répertoire" si la création n'a pas réussi.
  + Lister : liste tous les éléments fichiers et répertoire du répertoire courant.
  + Lister-D : liste tous répertoires du répertoire courant.
  + Lister-F : liste tous fichiers du répertoire courant.
  + Renommer <old name><new name> : modifie le nom d’un fichier ou d’un répertoire.
  + Cd<path> : prend le chemin et passe d’un répertoire a un autre. Aussi on peur naviguer avec le Path et naviguer vers les parents.
  + Rechercher <nom >: recherche toutes les ressources portant le nom spécifiés dans le système de fichiers sinon un erreur s’affiche.
  + Supprimer : supprime un fichier ou une ressource et tous ses descendants, le cas échéant.
  + Supprimer -f: supprimer un répertoire non vide , imprimer le résultat ("répertoire supprime avec succès ").
  + chemin : affiche l'emplacement de la ressource dans le système de fichiers en affichant le chemin absolue vers le répertoire courant.
  + Copier <nom><Path> : copie un fichier dans un autre répertoire avec le même nom.
  + Help -- : affiche toutes les fonctions/commande du système.
  + Quitter : met fin à l'exécution du gestionnaire de fichier

Code Source :

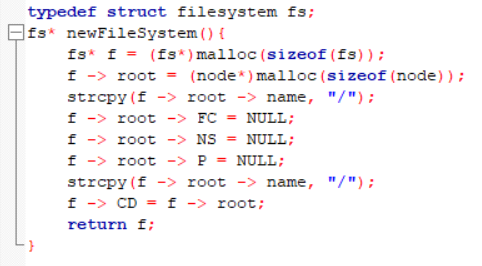
>> La structure d’un nœud d’arbre



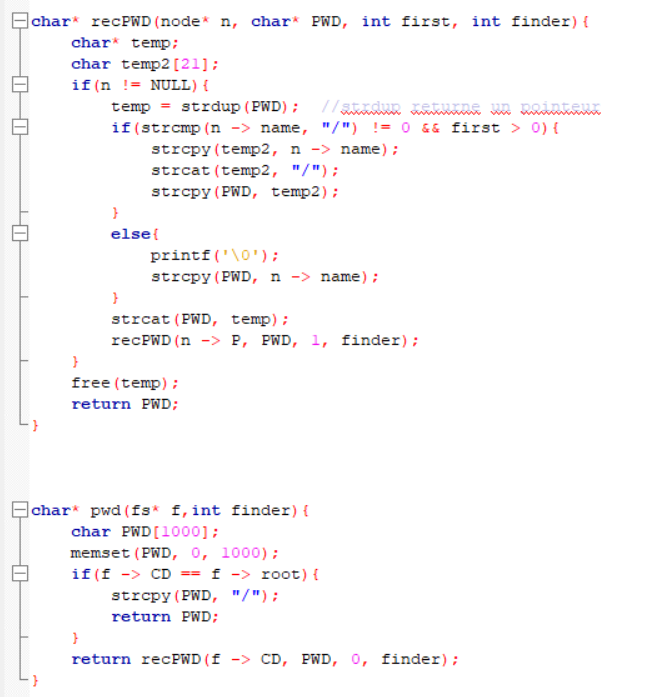
>>La structure de l'arbre d'un système de fichiers



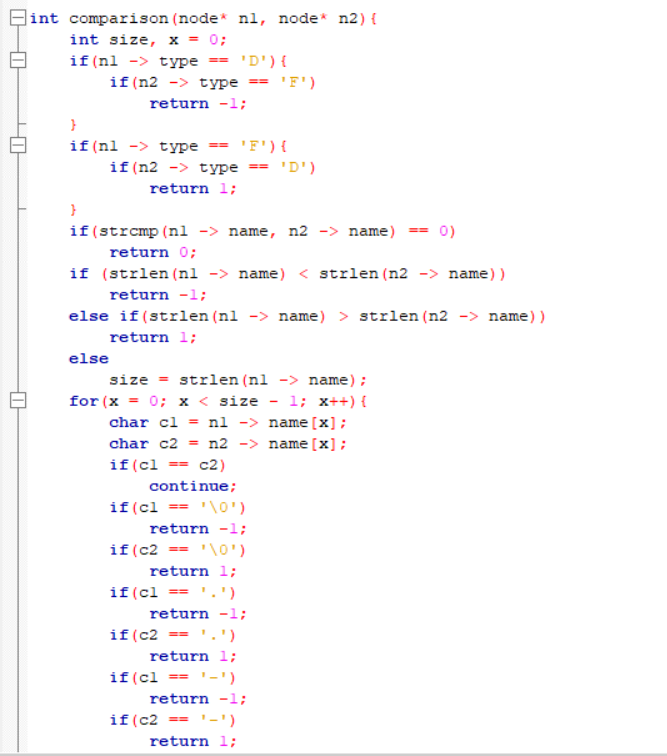
>>Initialiser l'arbre (création de la racine)

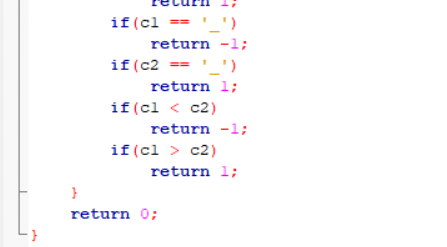


>>chemin absolu récursive

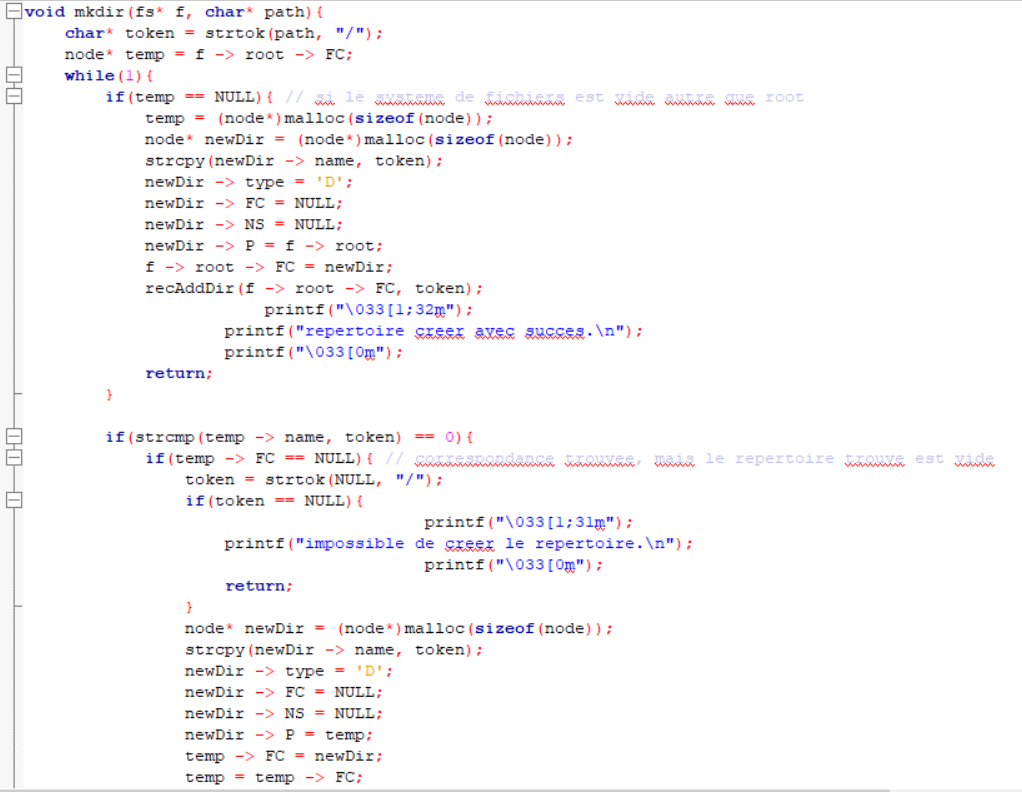


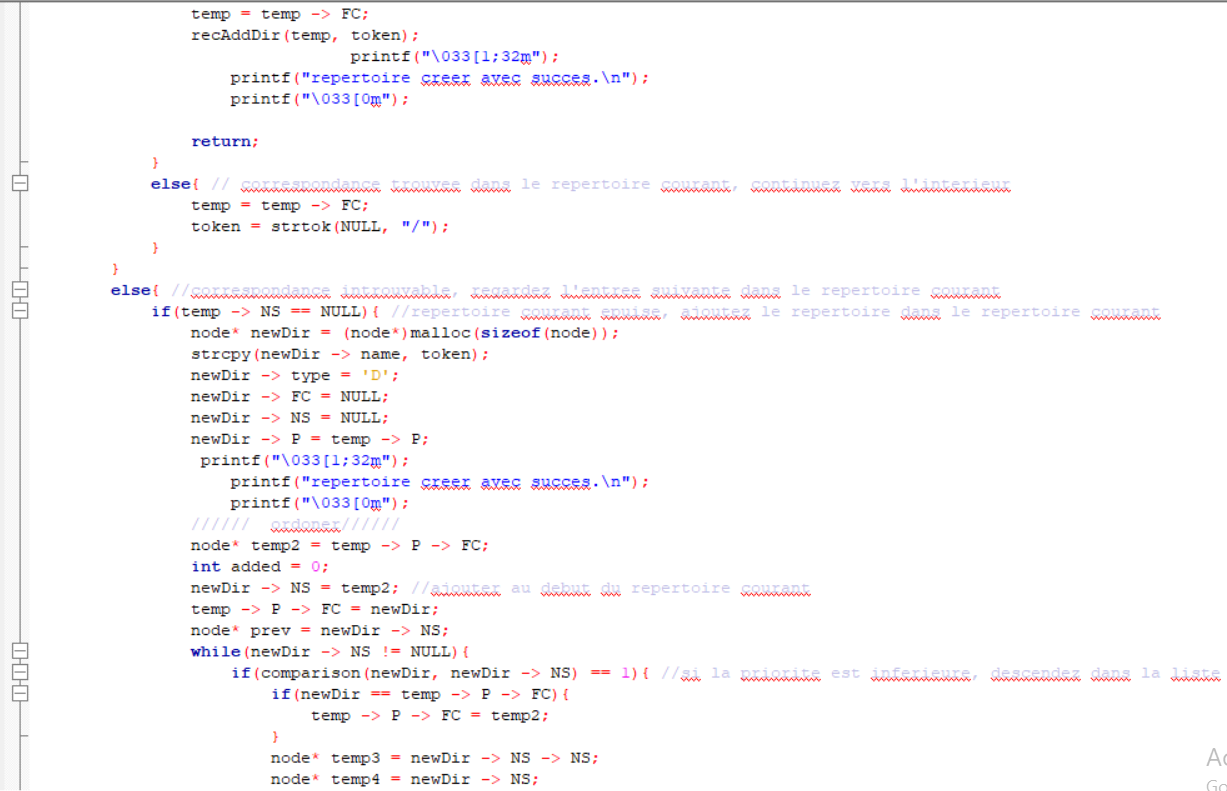
>>comparer 2 noeuds (type / nom / taille )

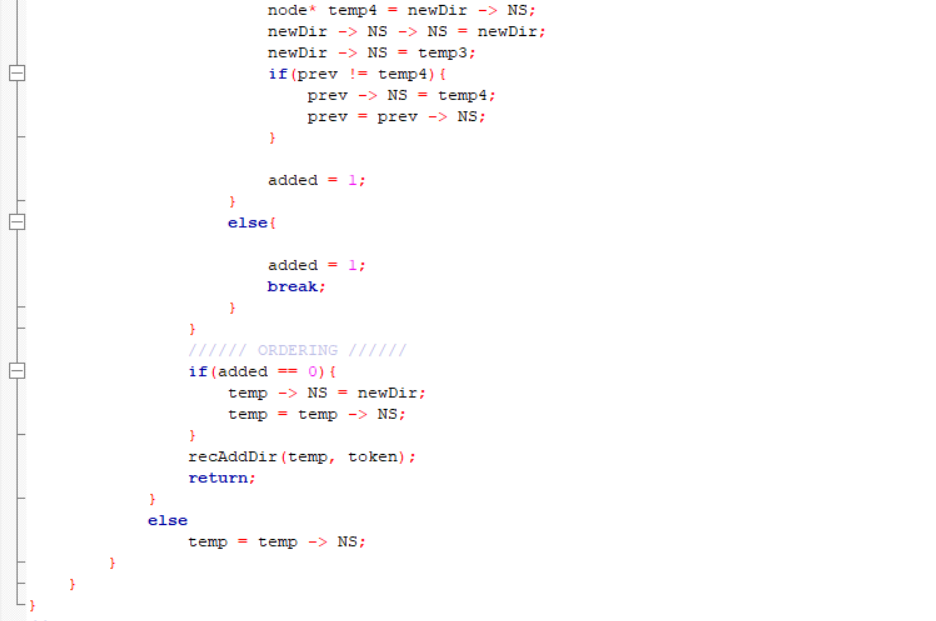




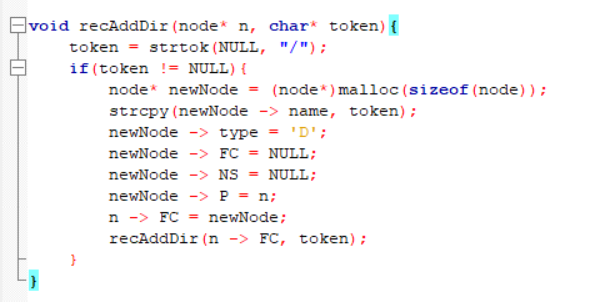
>>fonction pour créer un répertoire



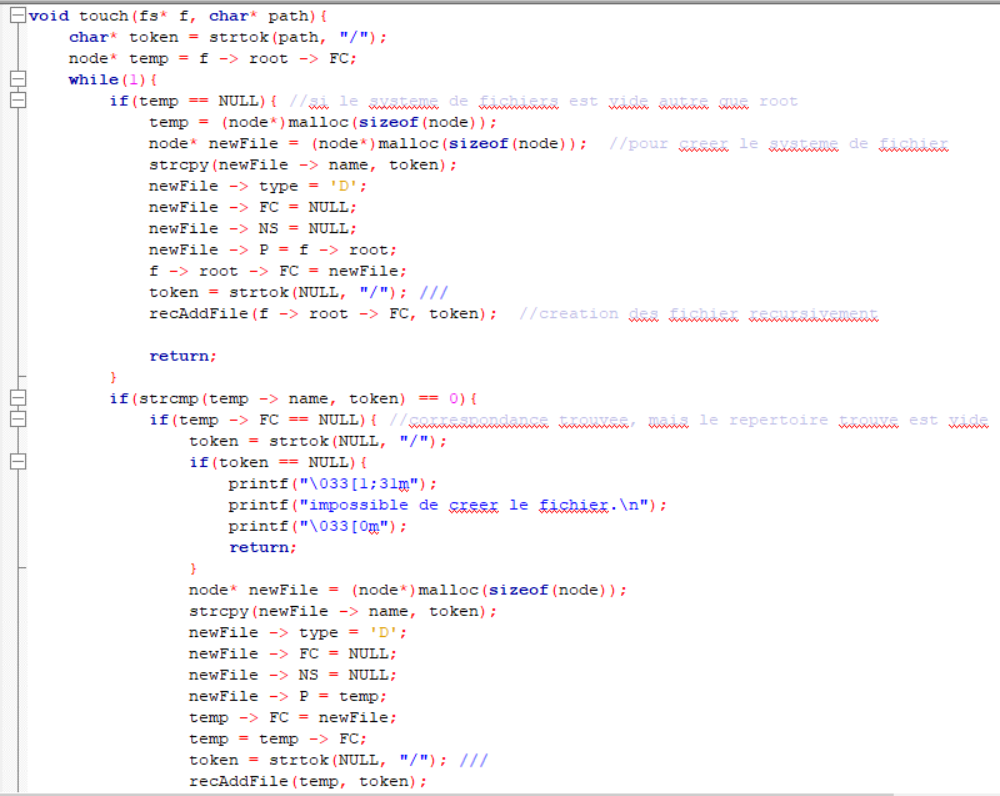


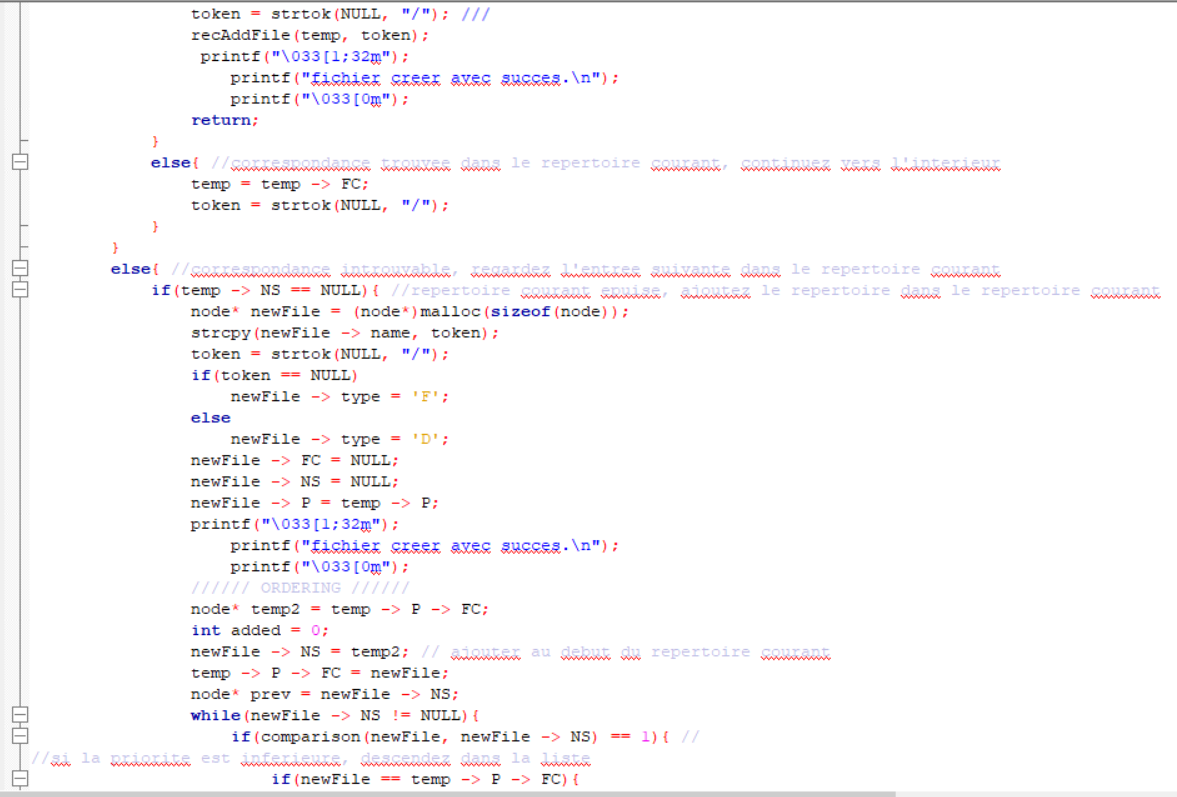


>>fonction de création des répertoires récursivement

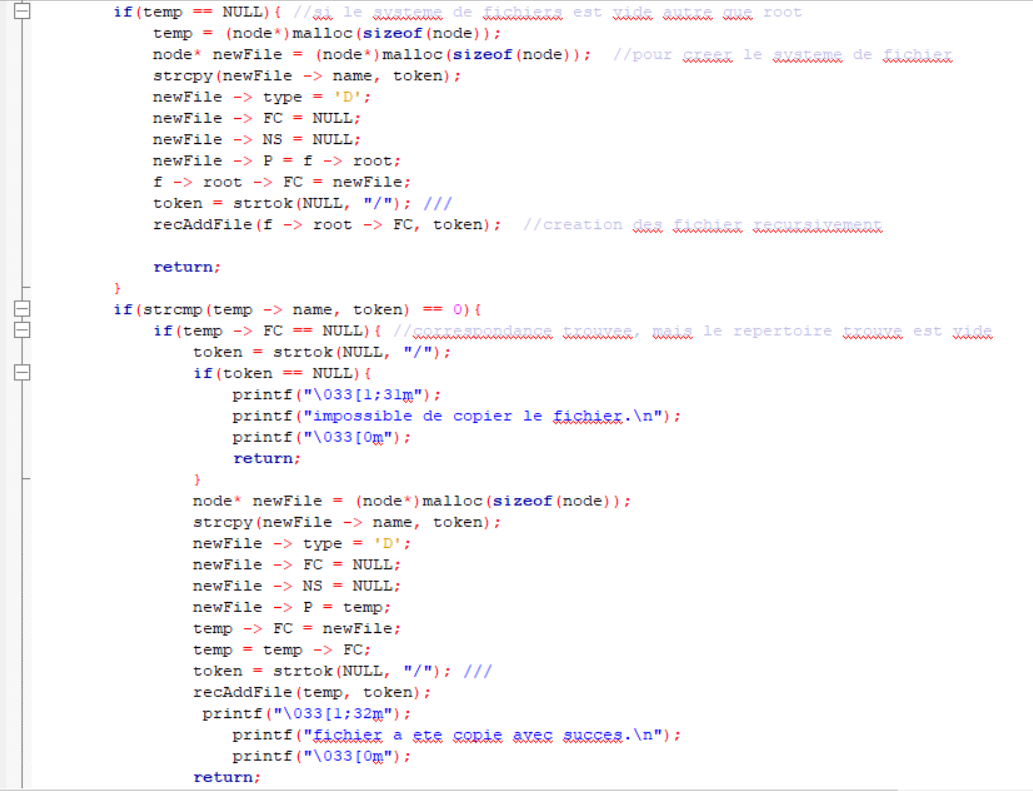


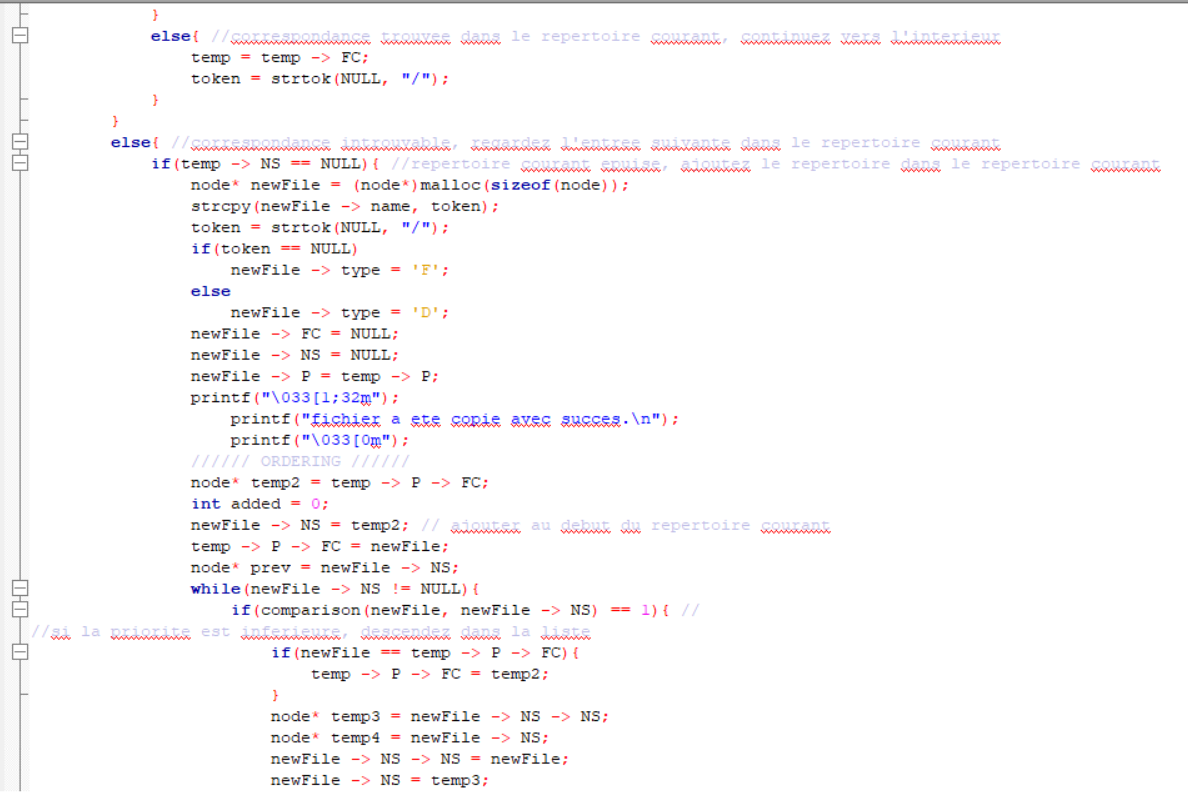
>>fonction pour créer un fichier

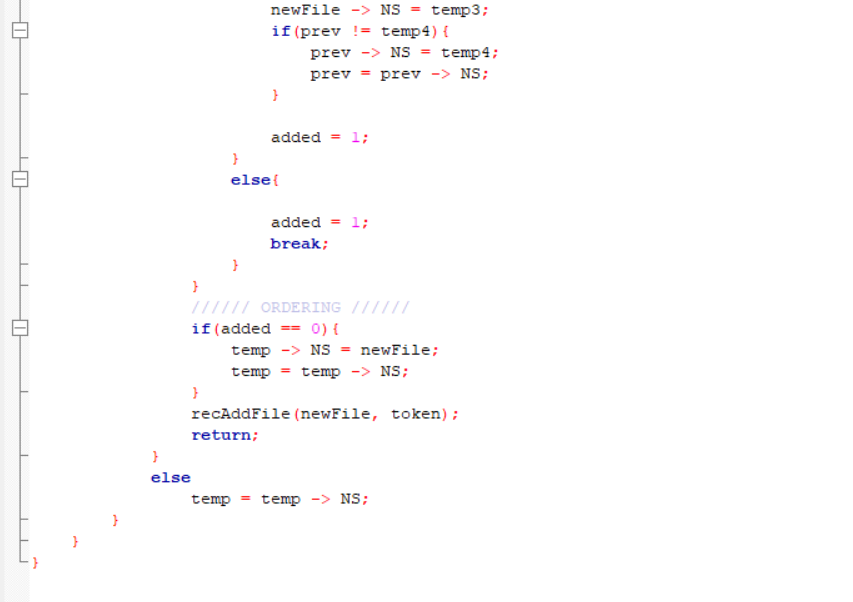








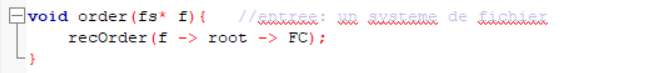




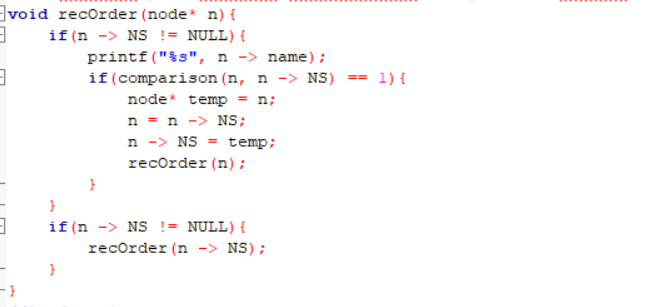
>>la fonction de création des fichiers récursivement



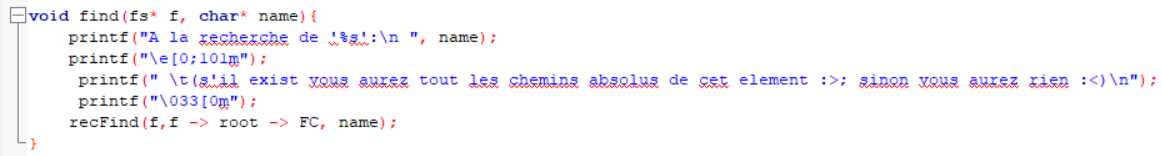
>>la fonction pour ordonner le system de fichier



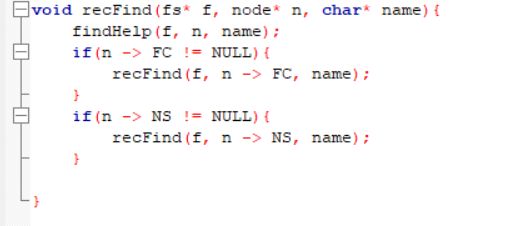
>>la fonction pour ordonner récursivement le system de fichier



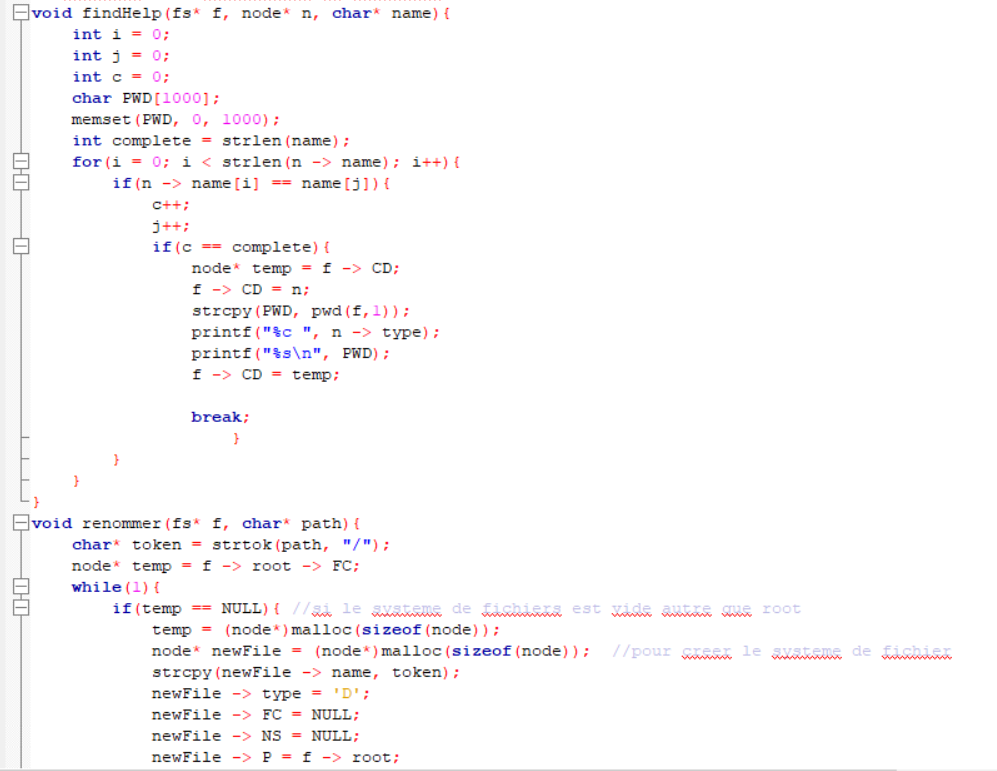
>>la fonction rechercher d'un fichier ou répertoire

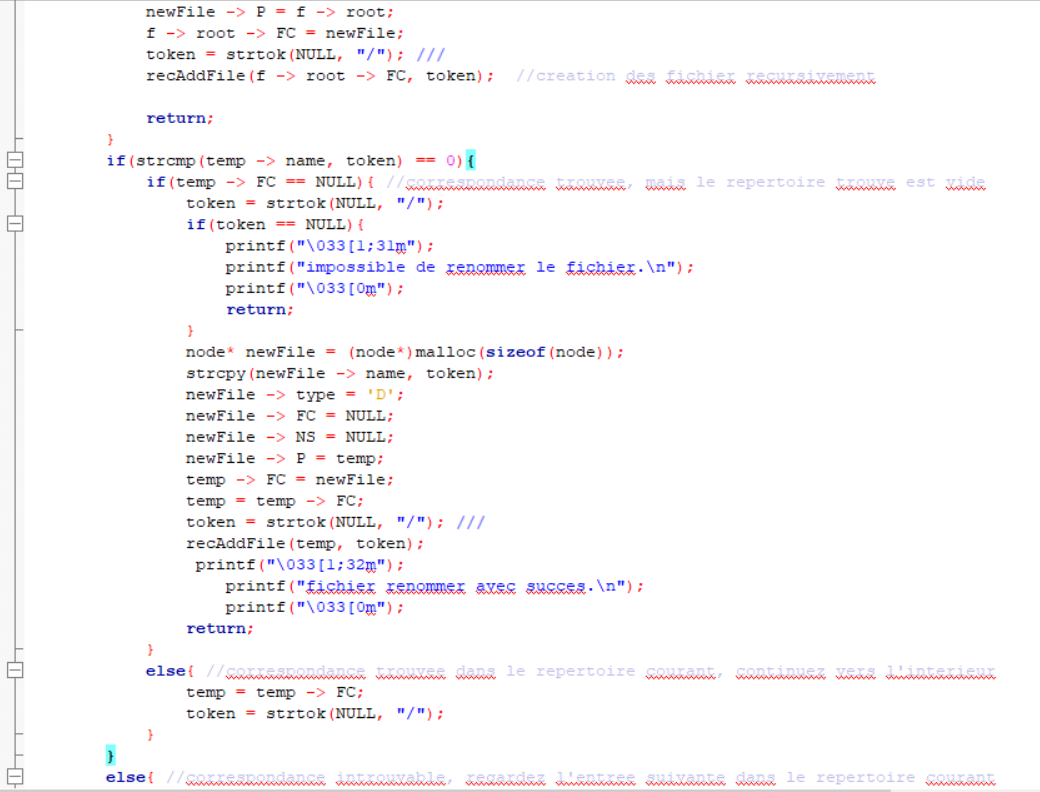


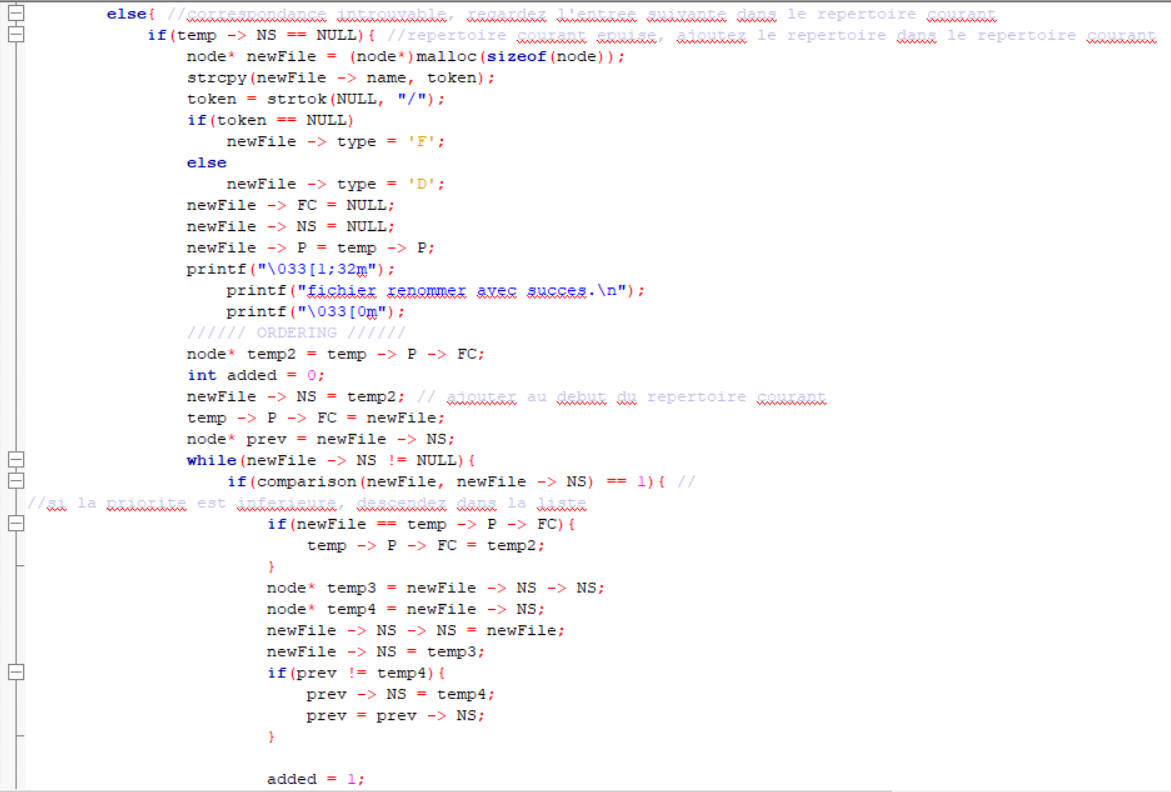
>>la fonction rechercher recursivement d'un fichier ou repertoire

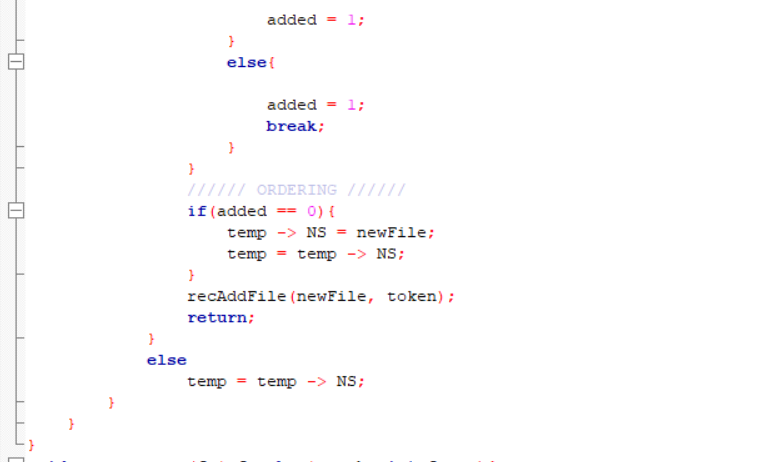


>>fonction pour l'exécution de la recherche récursive

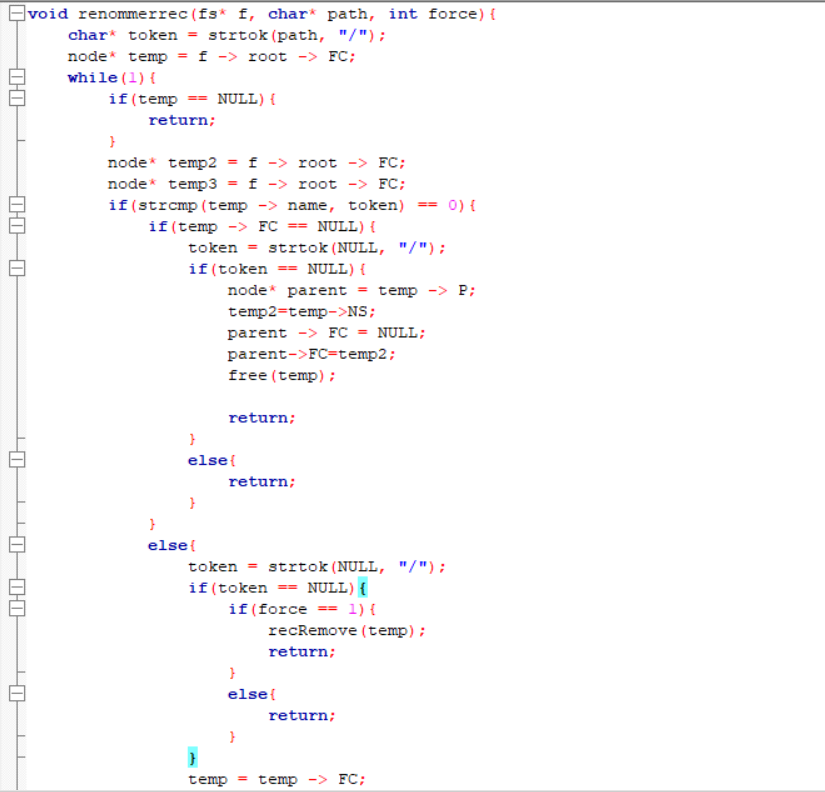


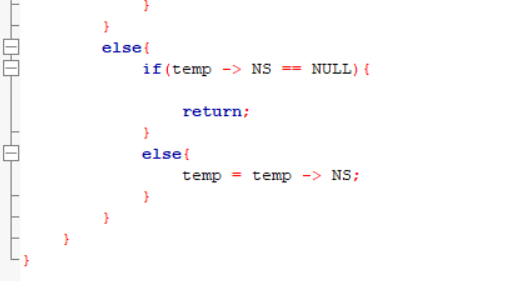




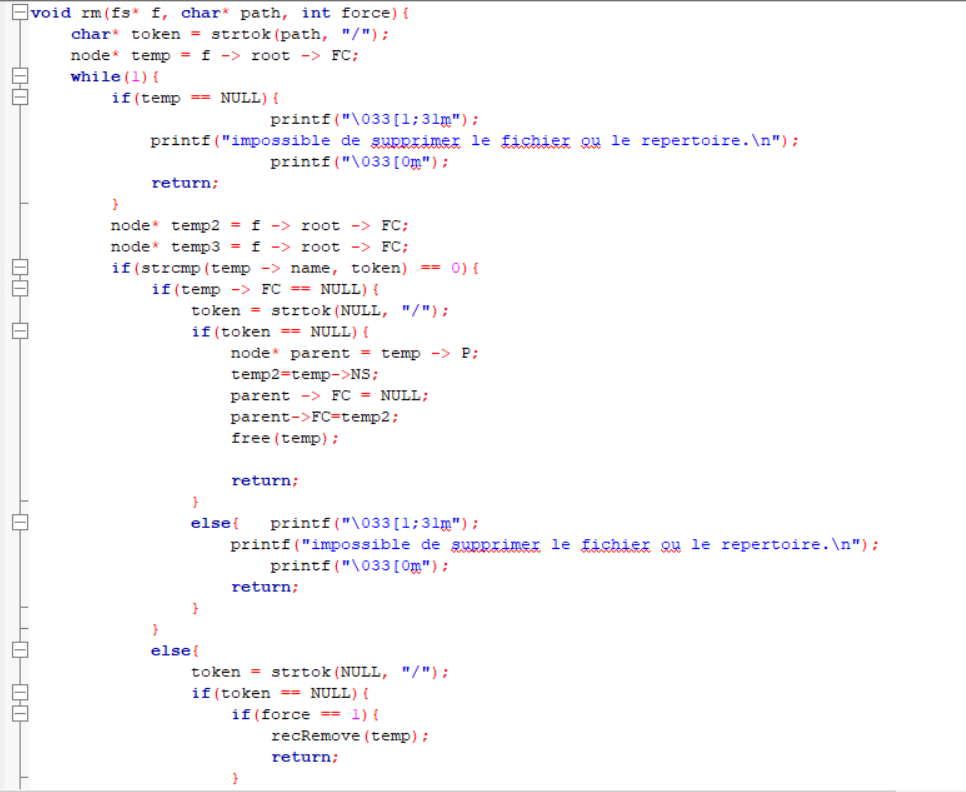


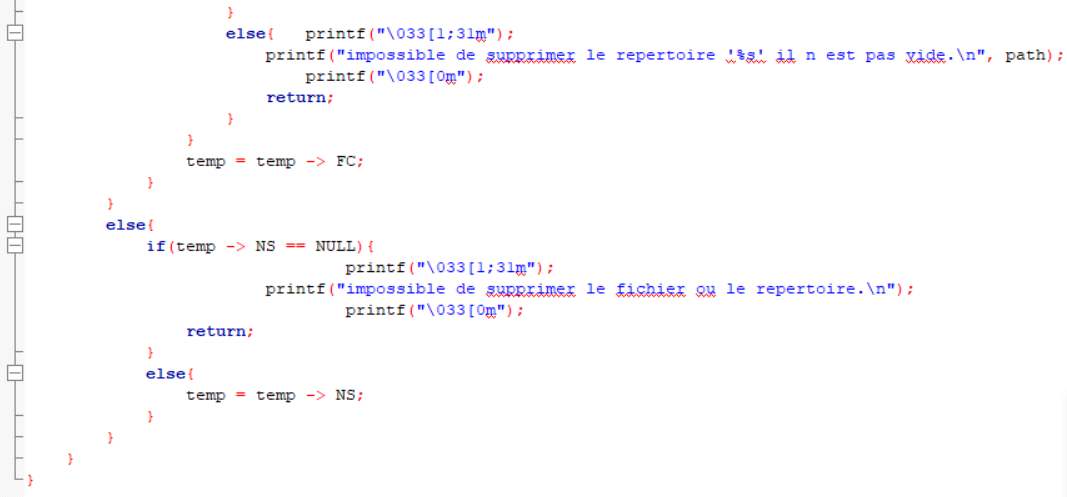
>>La fonction renommer



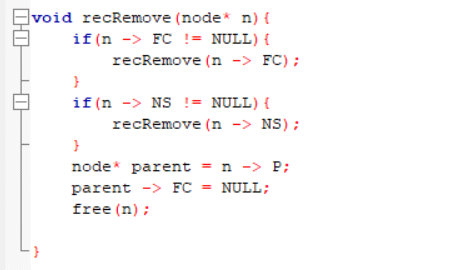


>>supprimer un fichier ou un répertoire

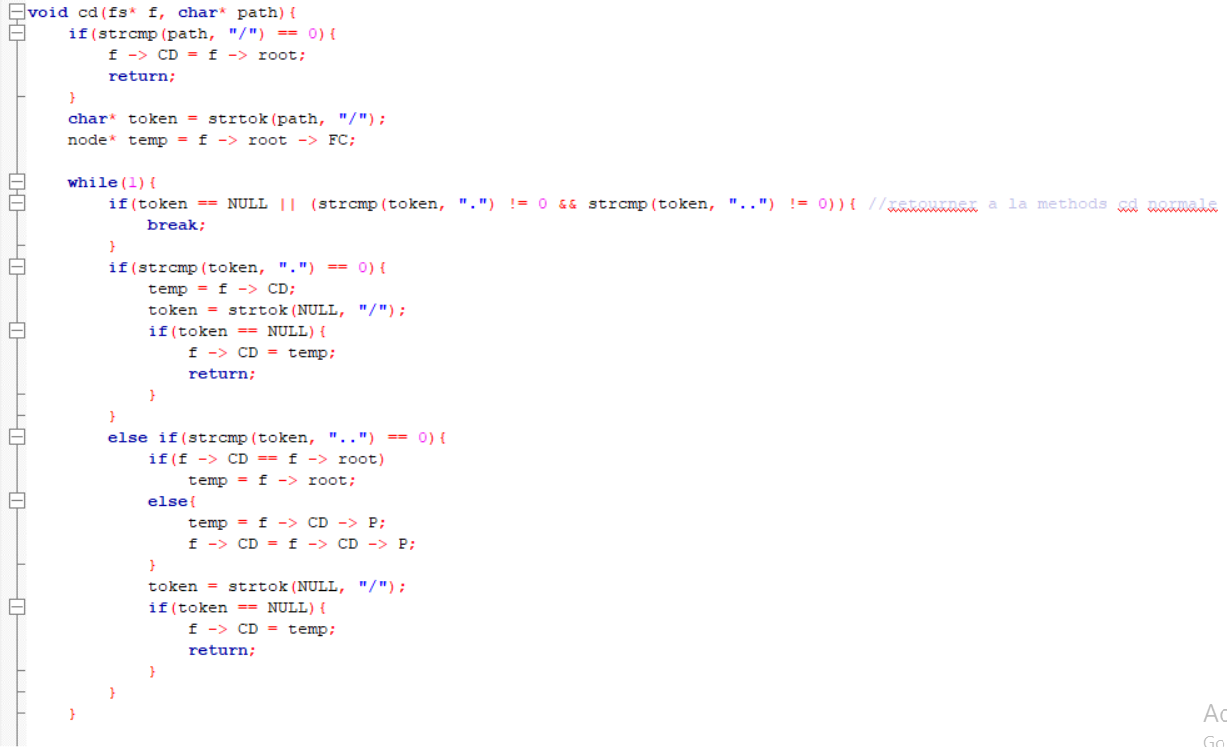


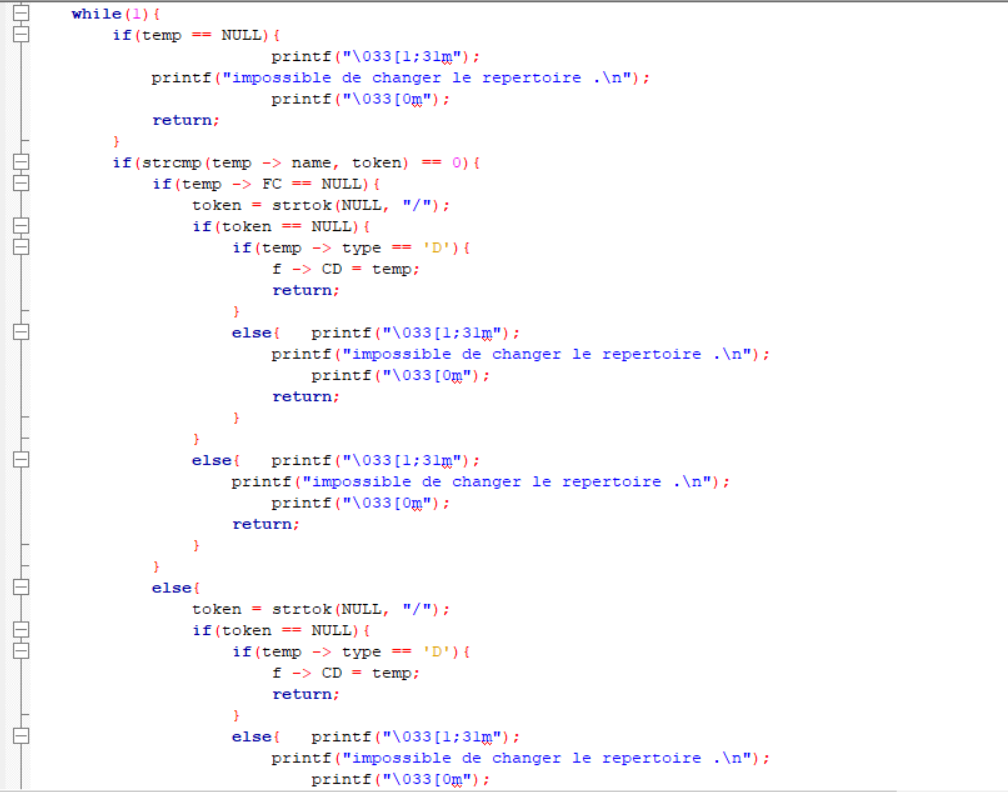


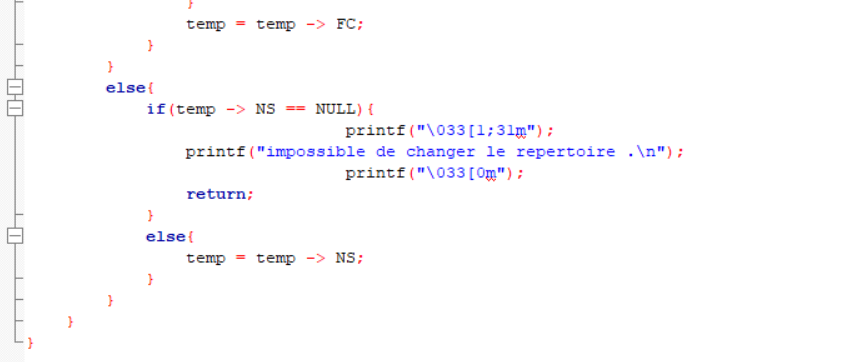
>>la fonction pour supprimer un élément récursivement



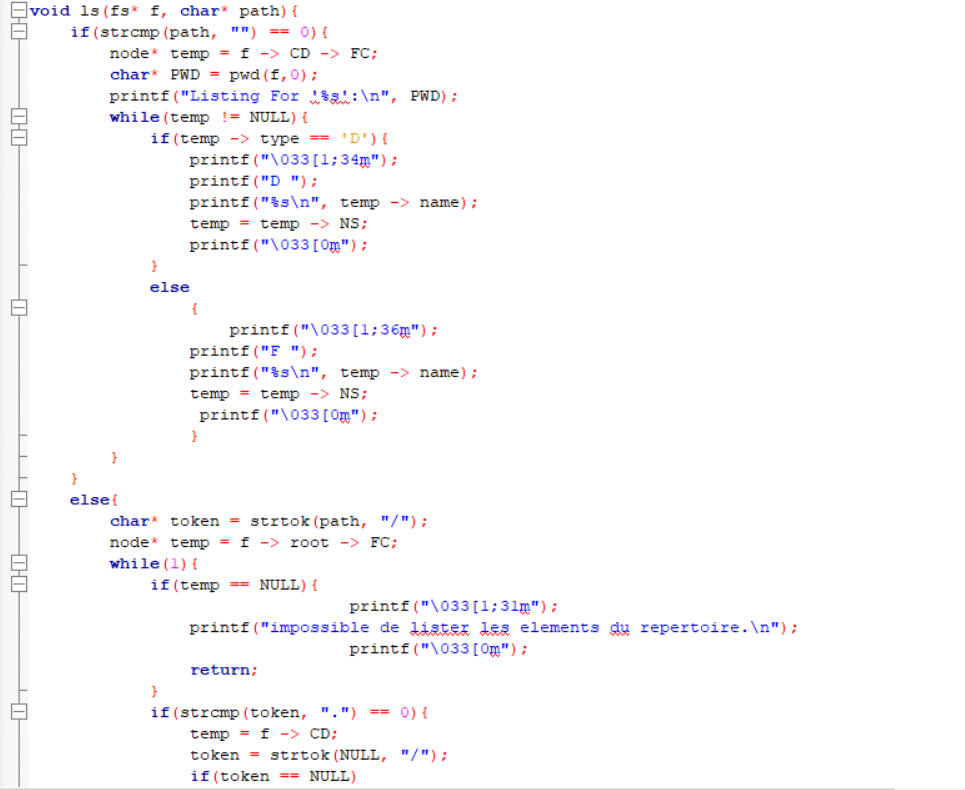
>>naviguer entre les répertoires

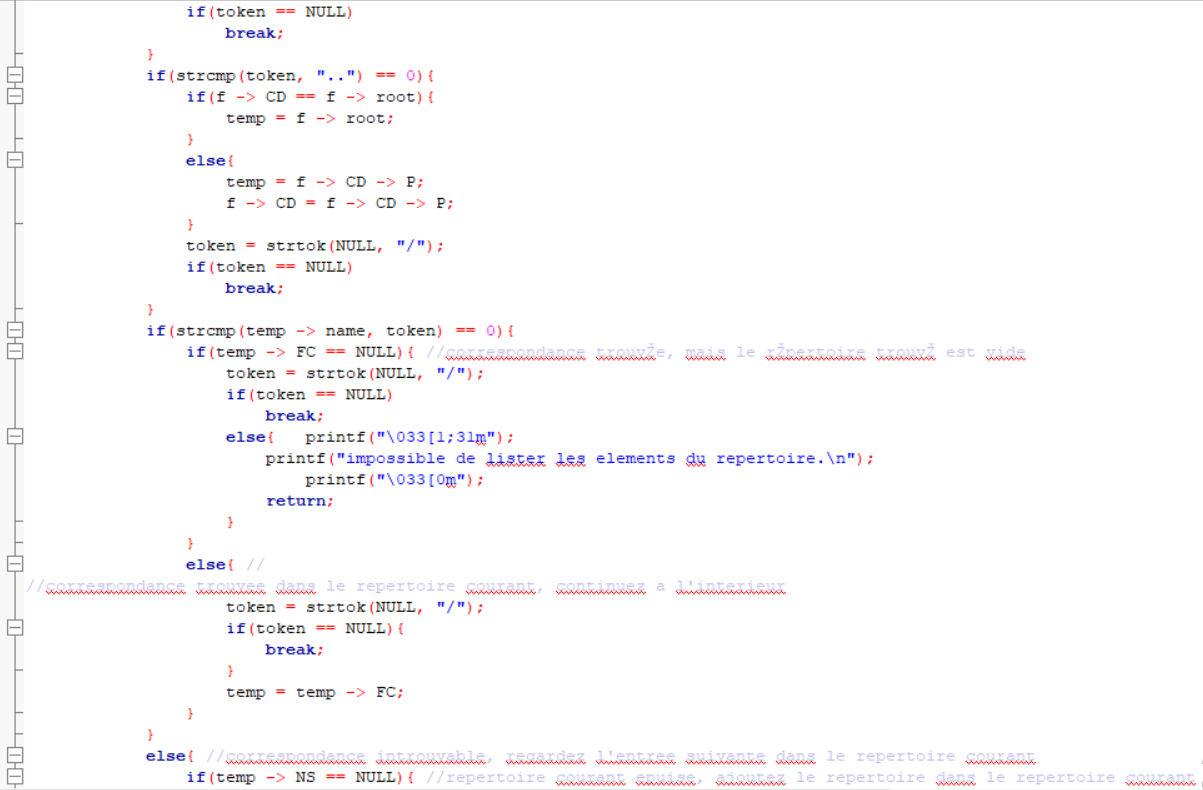


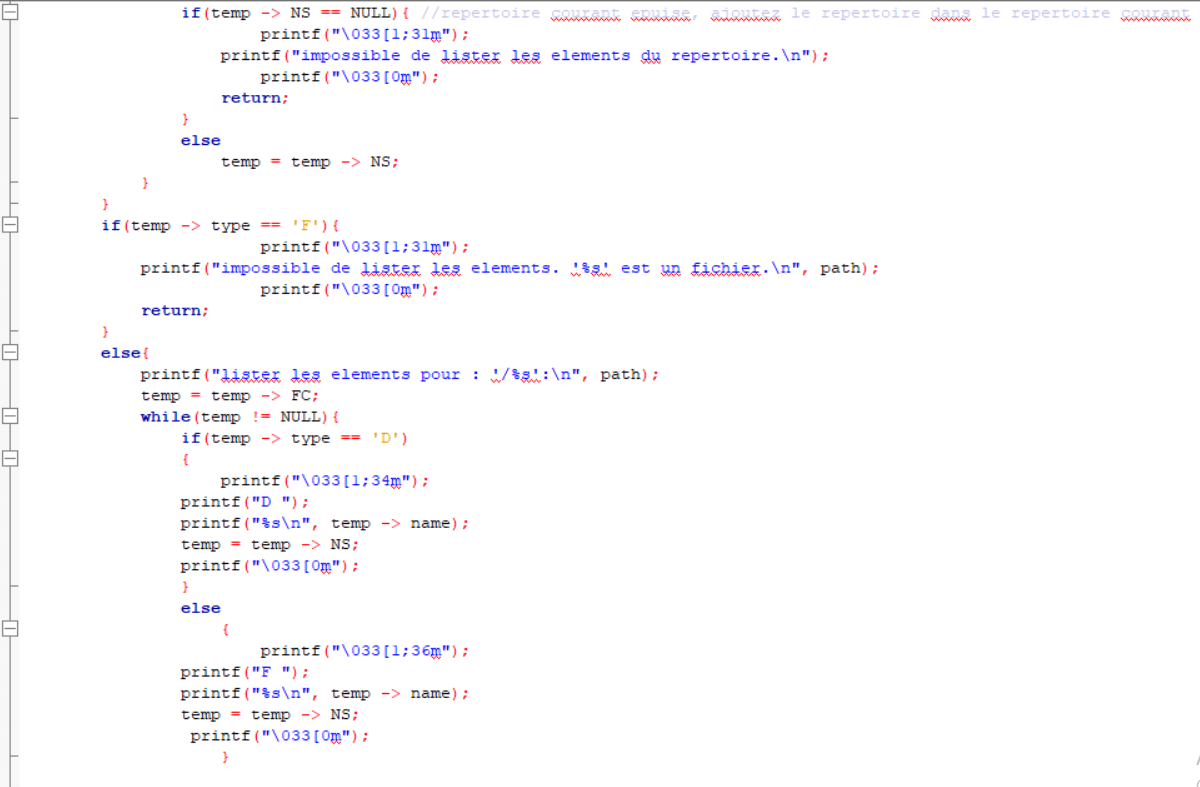


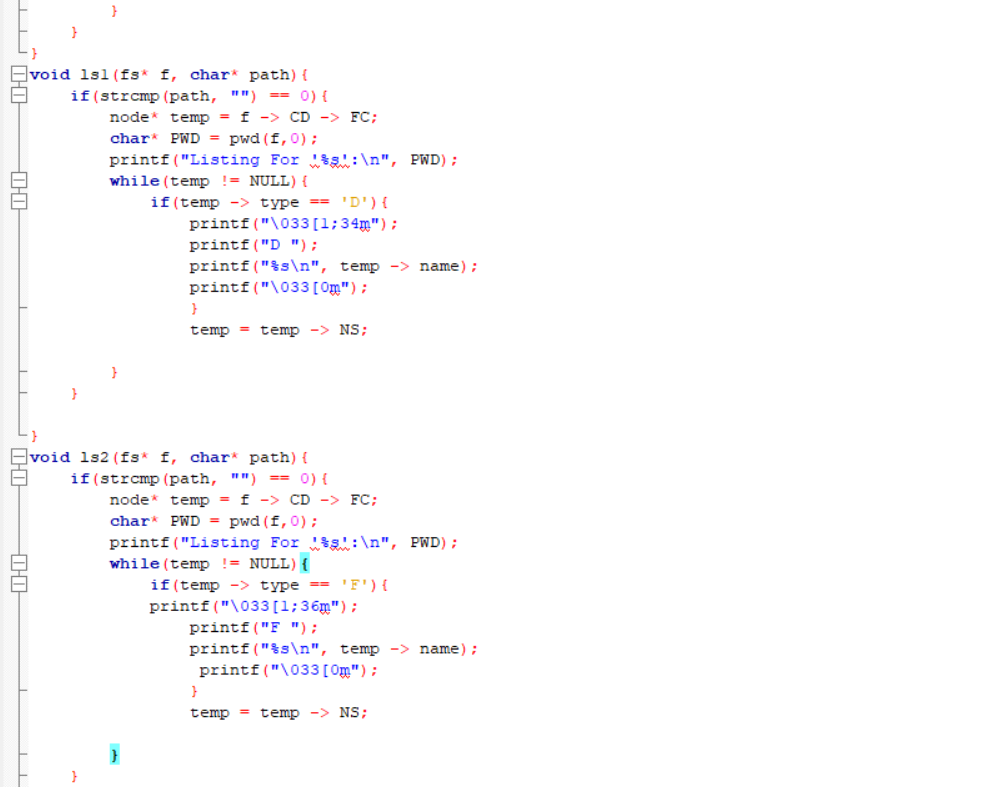


>>lister le contenu d’un répertoire

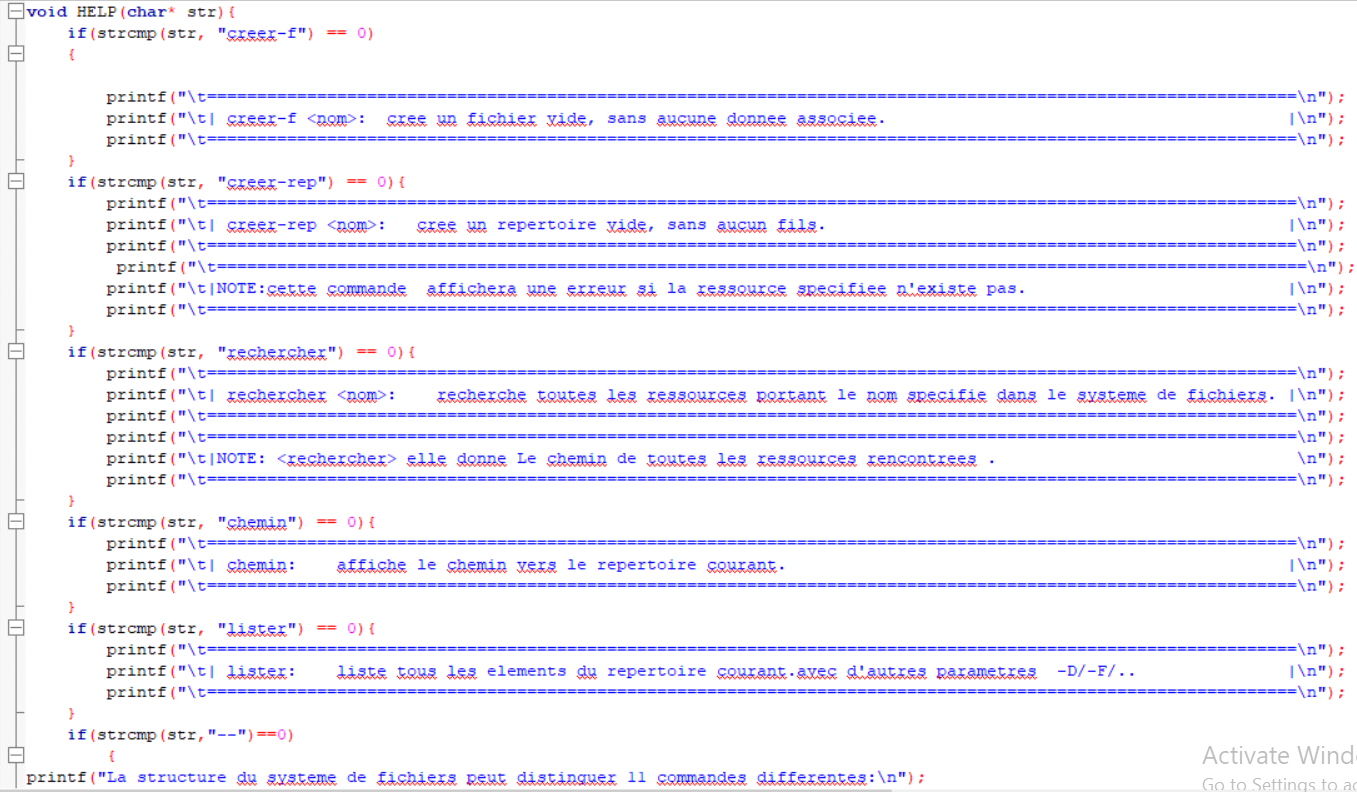


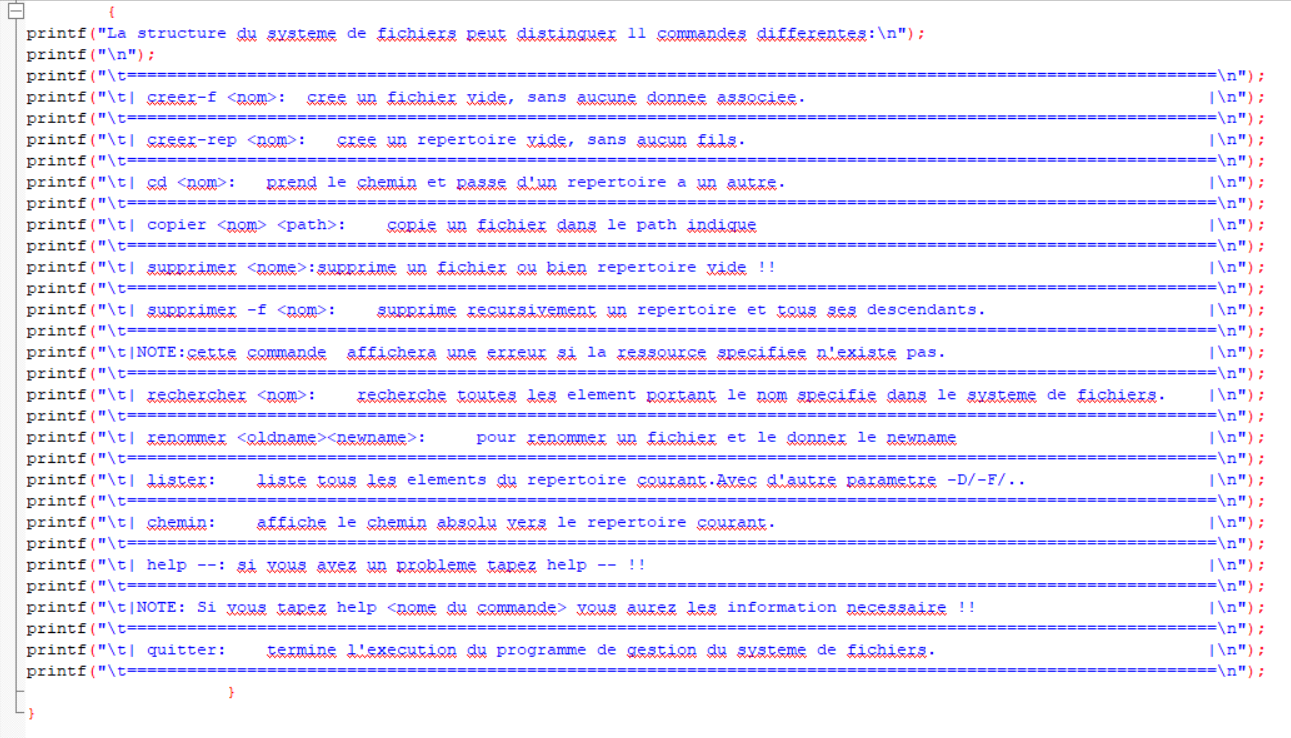




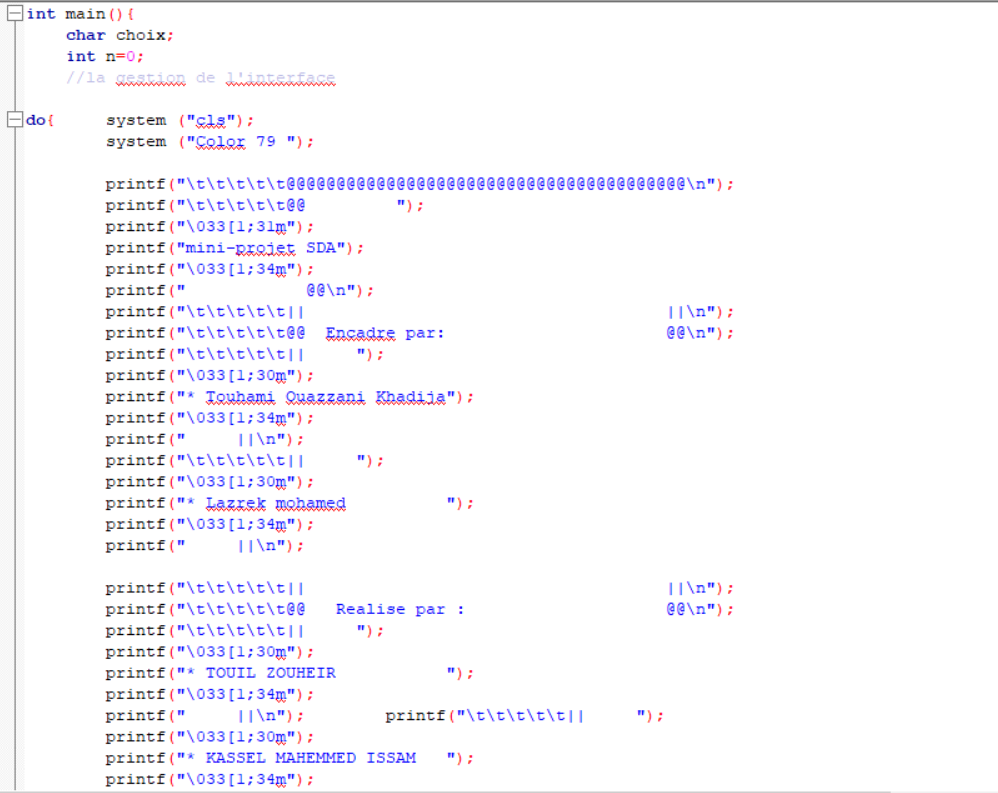


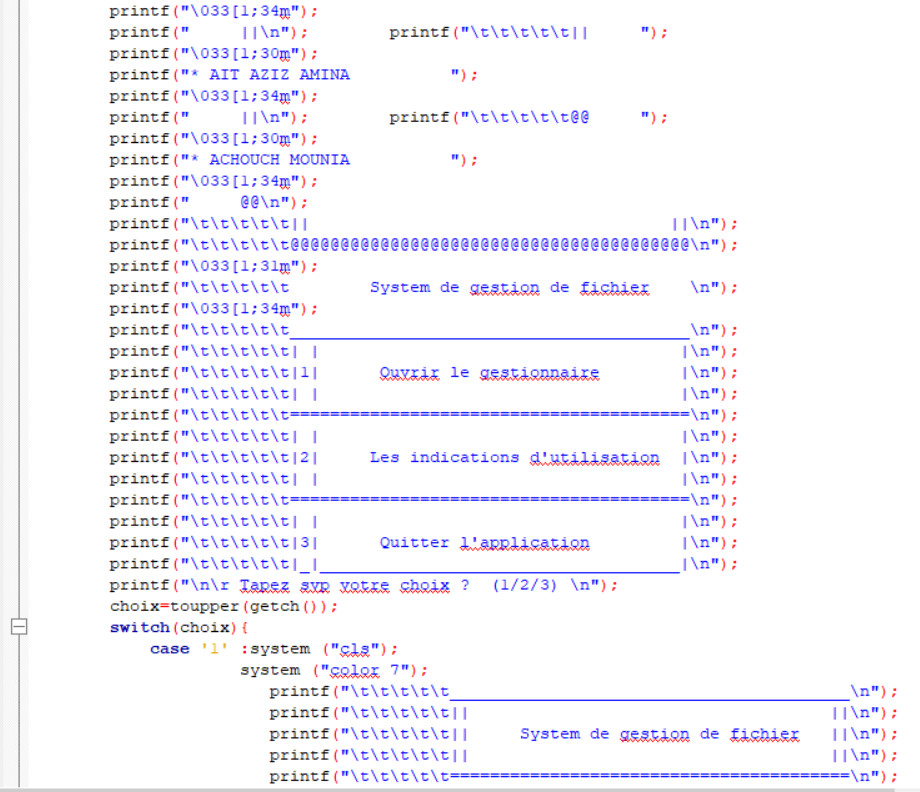
>>la fonction HELP



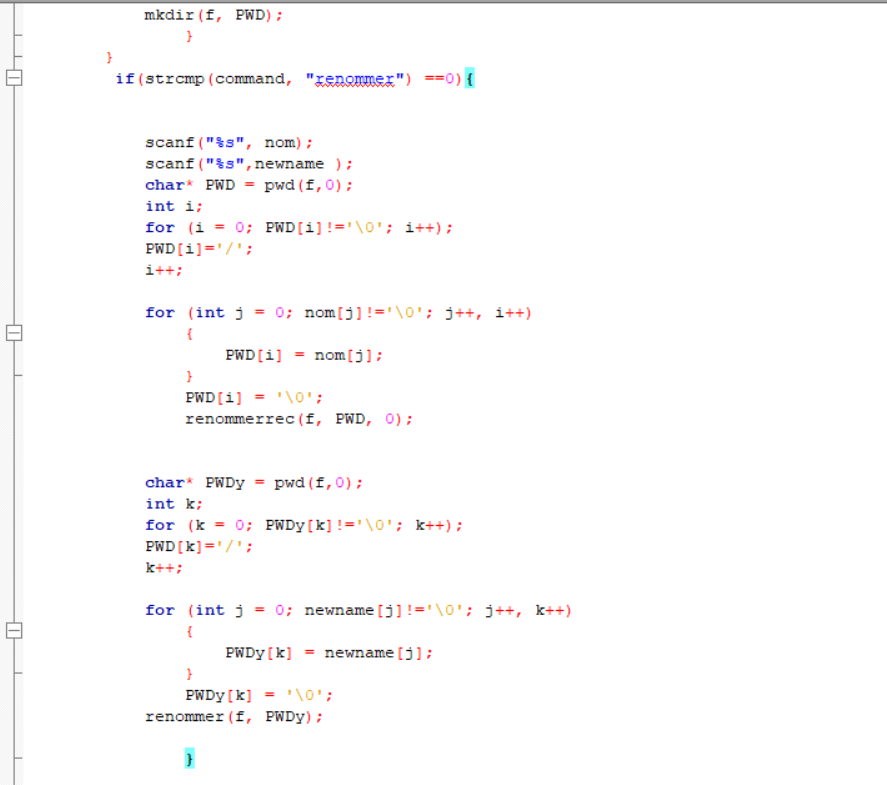


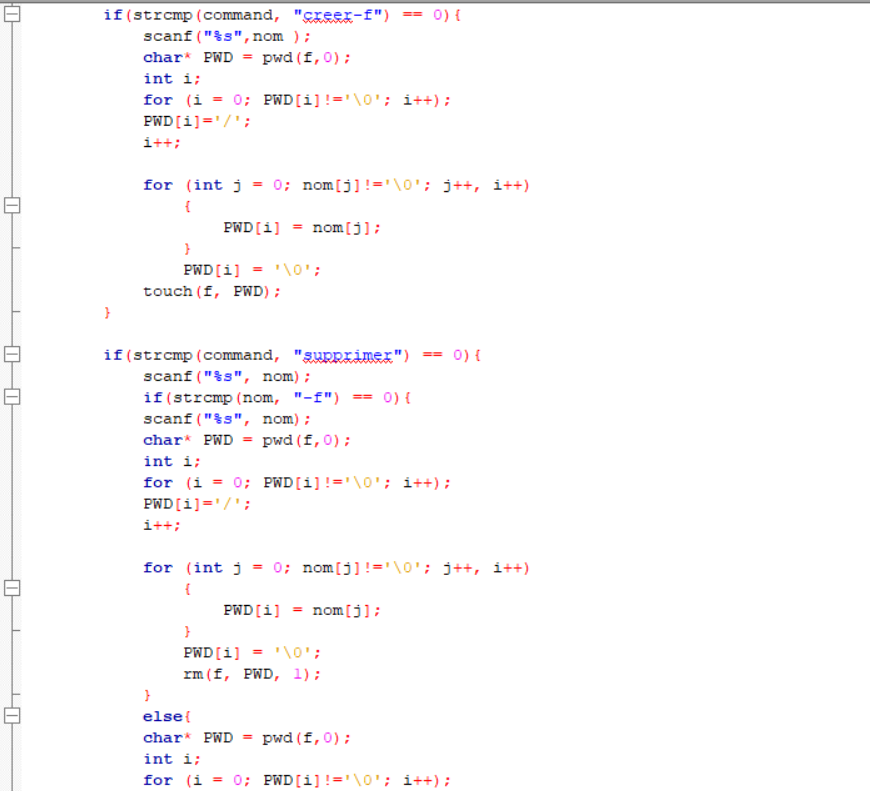
>>la fonction PRINCIPALE main

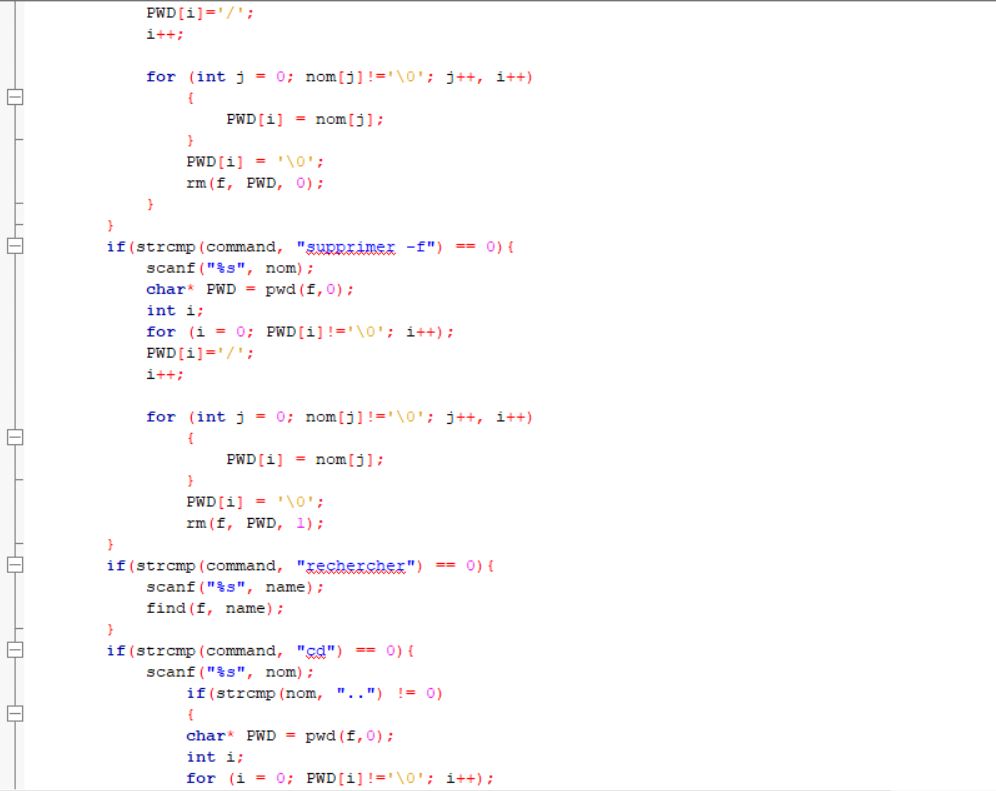


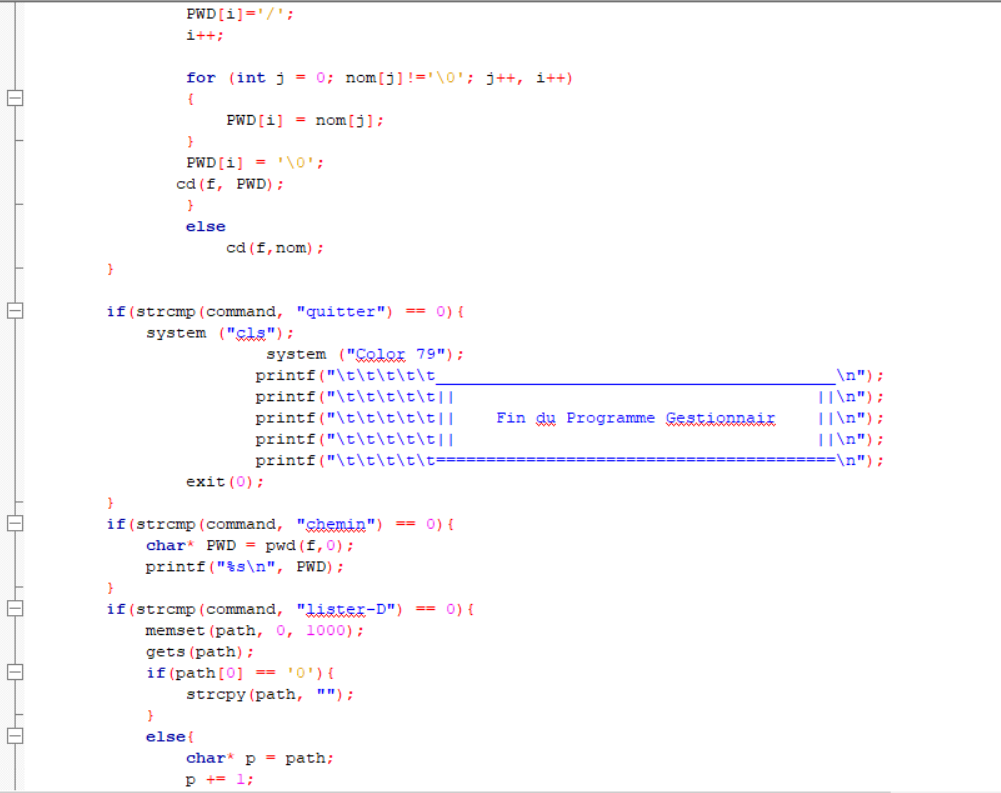


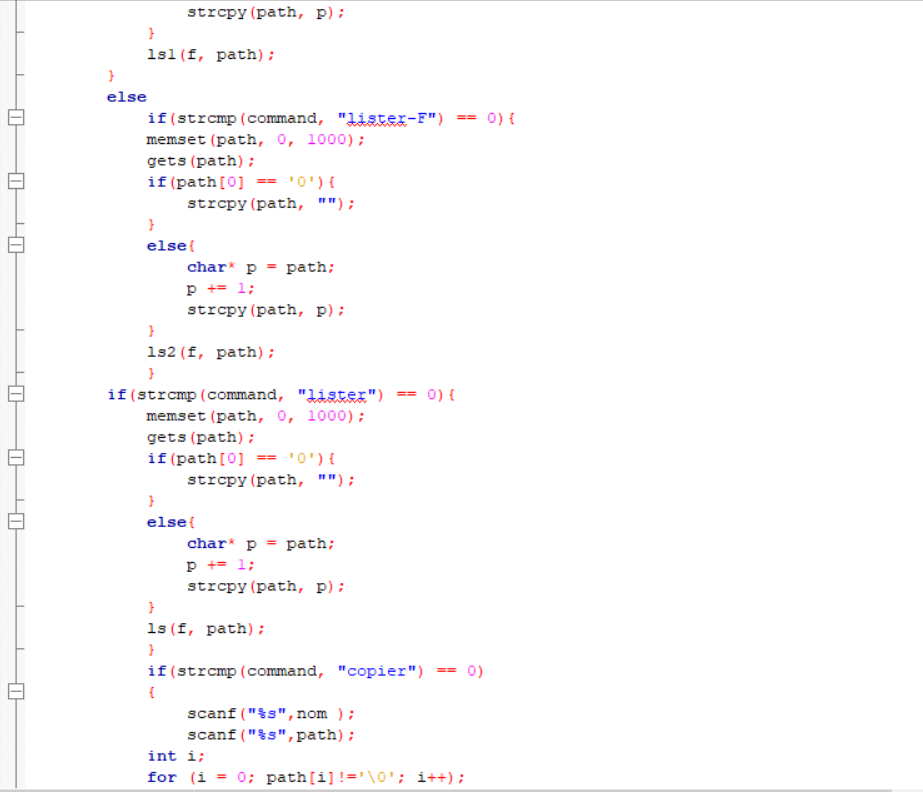


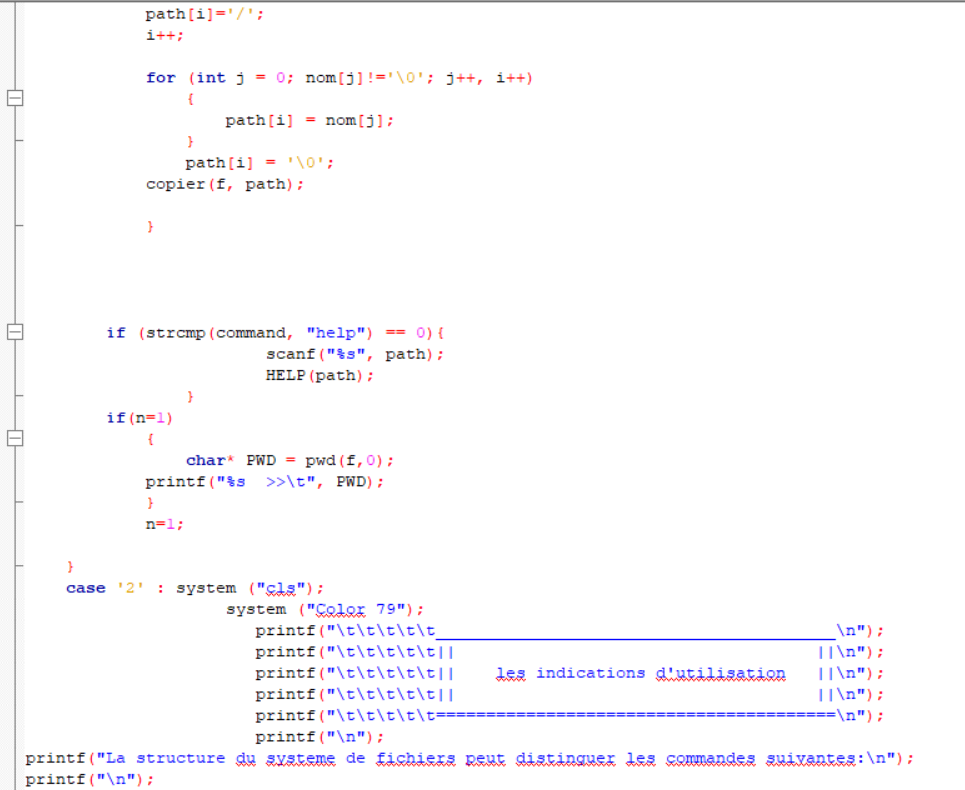


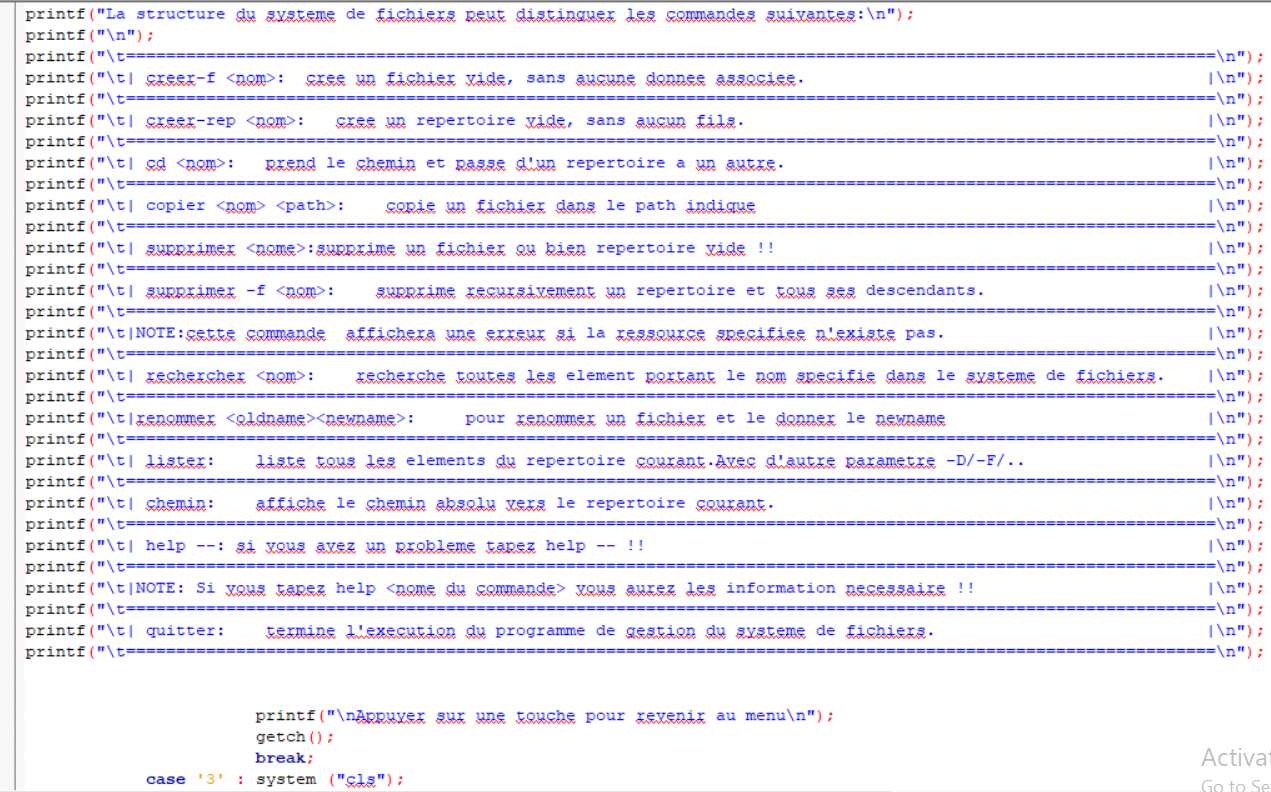


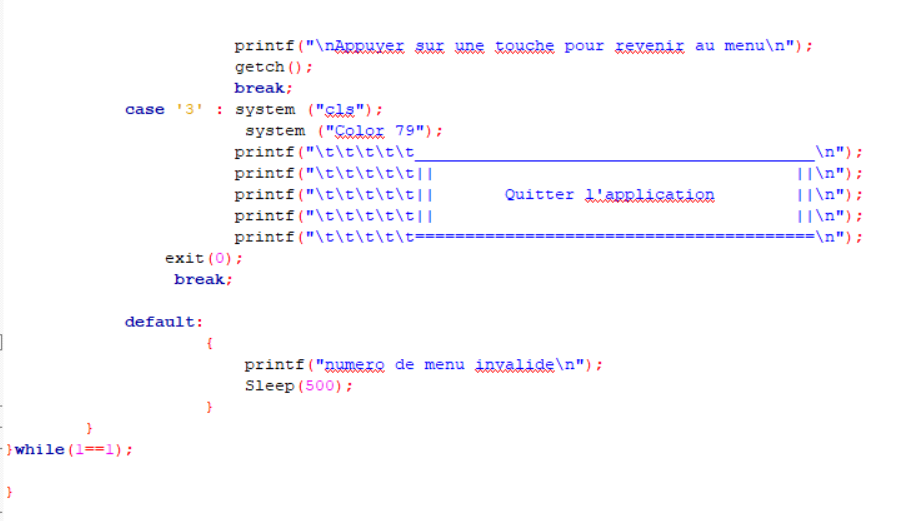












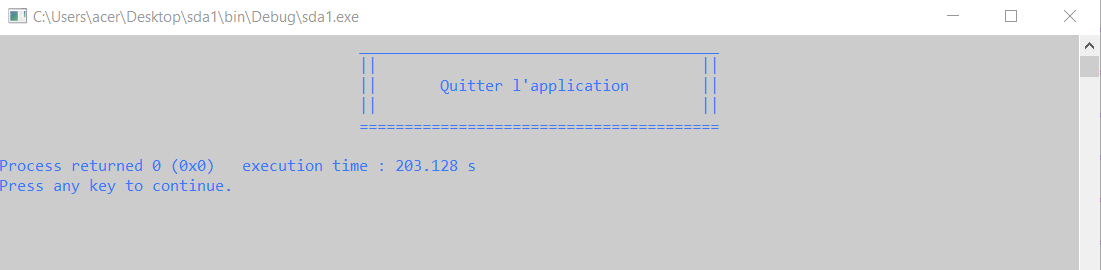
Exécution du programme :

>>L’interface

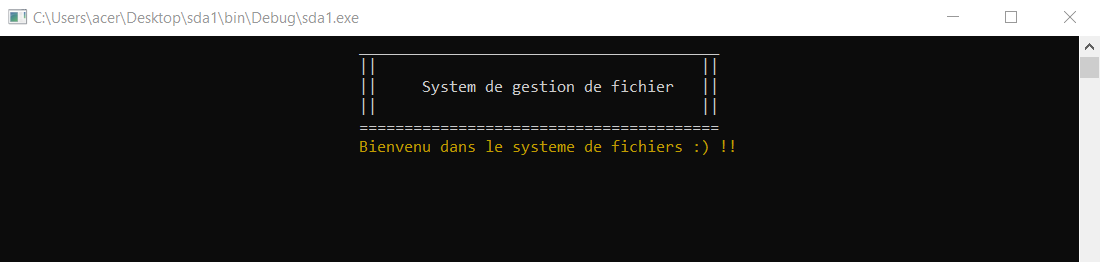


Selon le choix : Si on tape 2 :

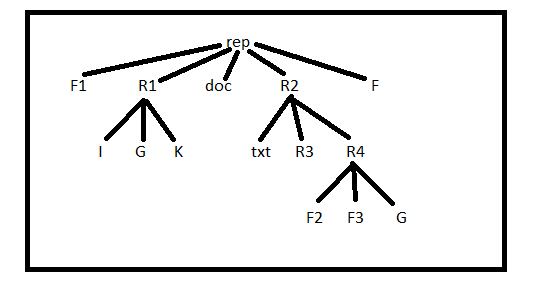
Si on tape 3 :



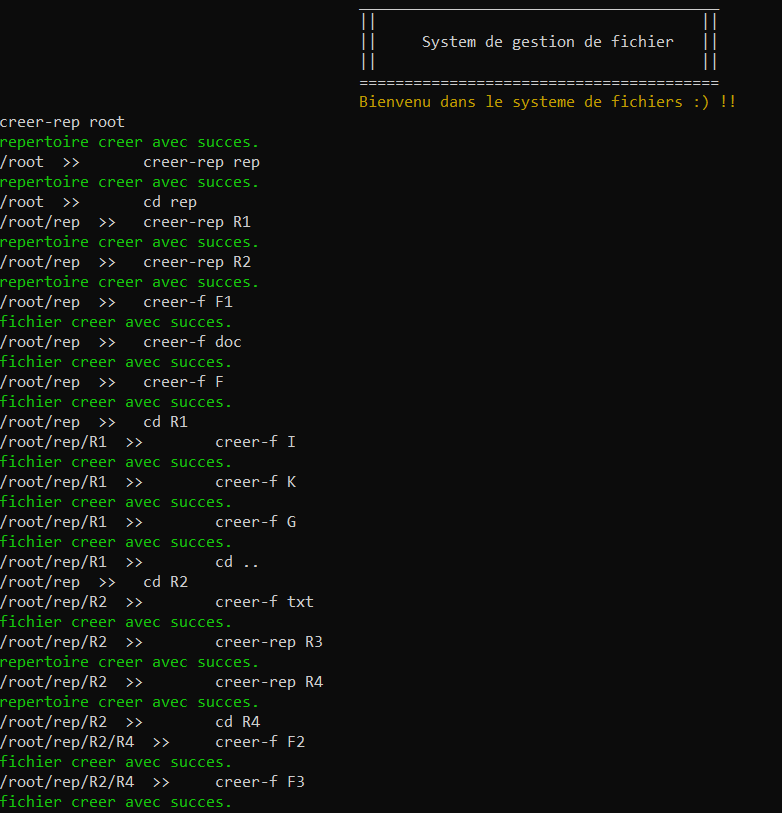
Si on tape 1 :



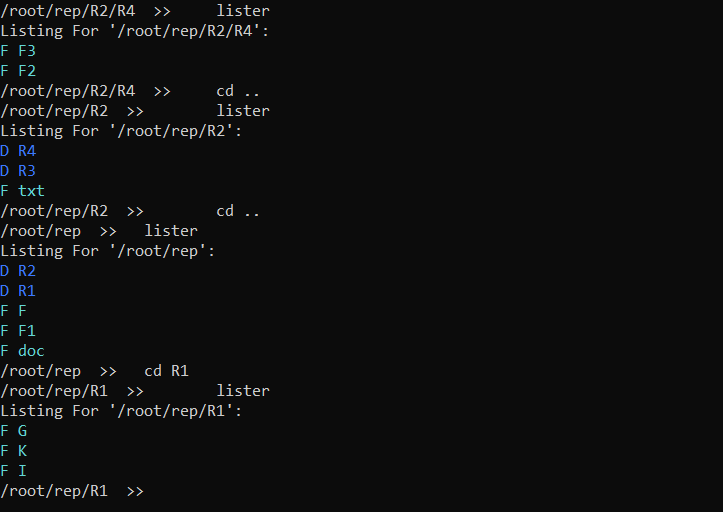
Pour l’exécution on va travailler sur l’arbre suivante :



>>Création :



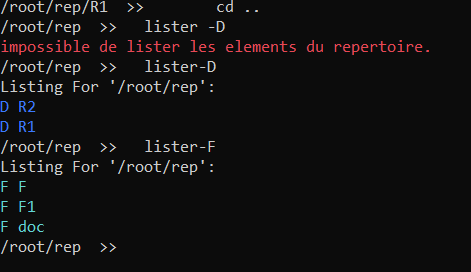
>>Lister les répertoires et les fichiers :



>>Copier le fichier G dans le répertoire R4 :

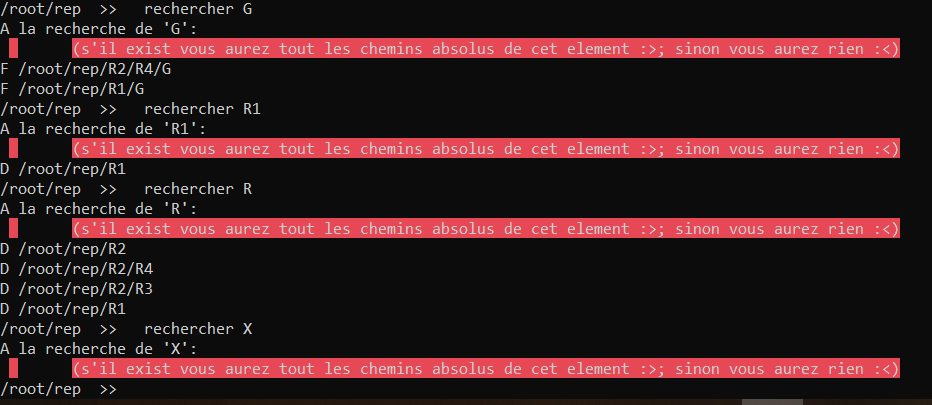


>>Lister répertoire/Lister fichier :



>>Rechercher :

Le cas où le répertoire ou le fichier existe il affiche le chemin absolue ,sinon un message d’erreur s’affiche.

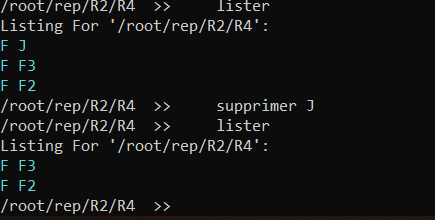


>>Renommer :

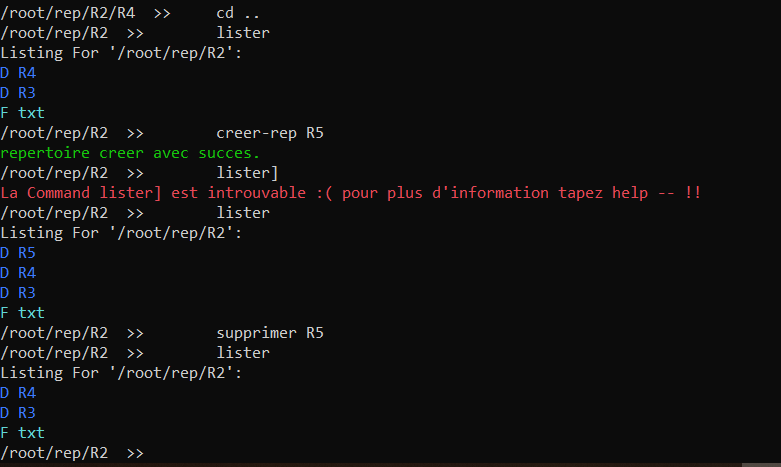
Renommer le fichier G du répertoire R4 en J



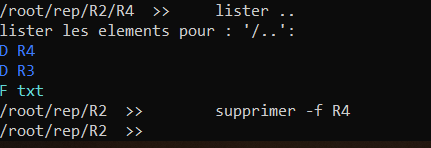
>>Supprimer fichier :



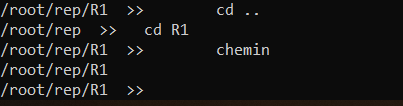
>>Supprimer répertoire vide :



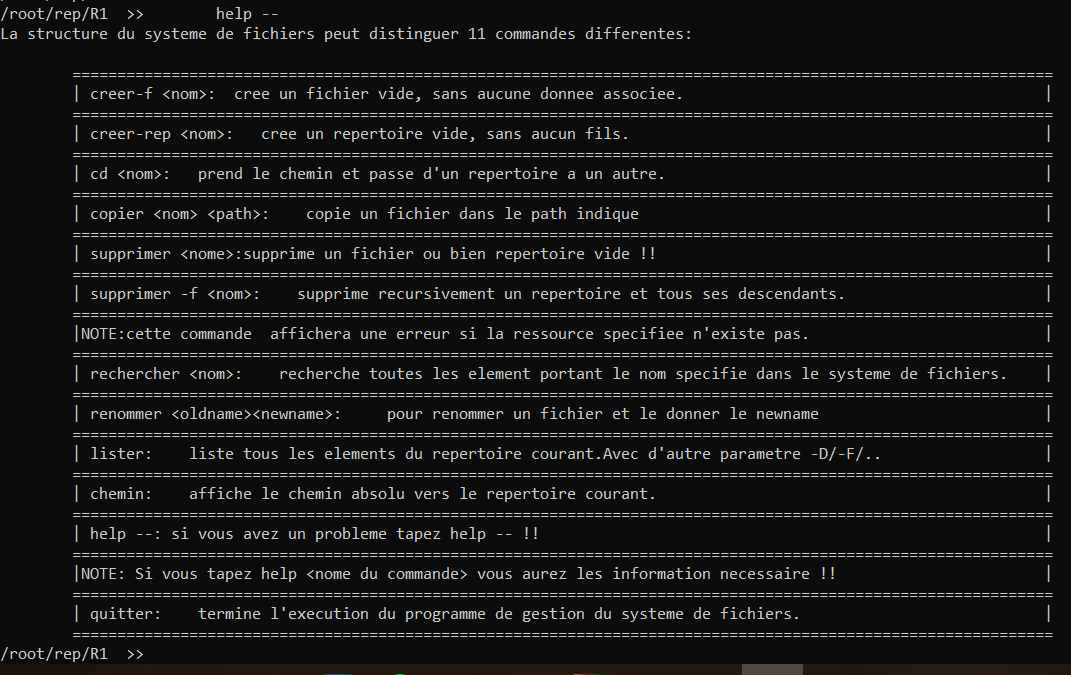
>>Supprimer répertoire non vide :



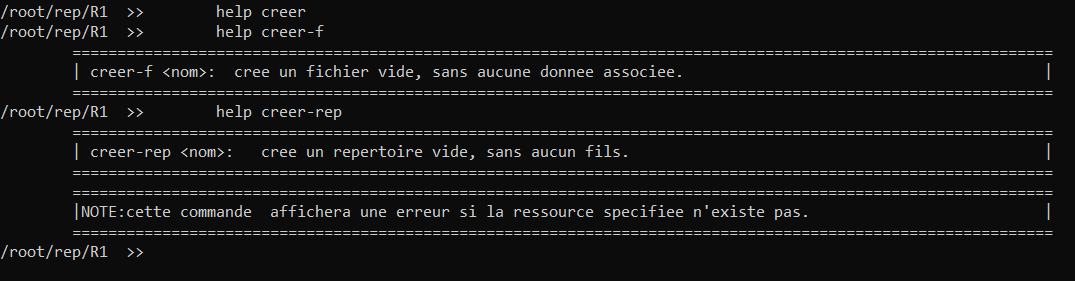
>>Afficher le chemin absolu :



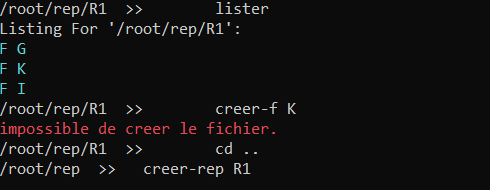
>>Afficher les fonctions du système :

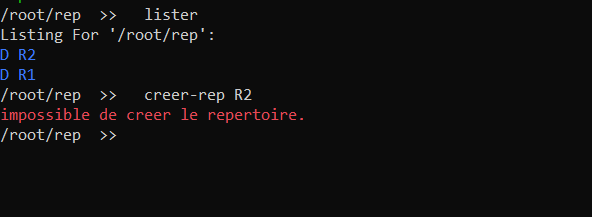


>>Afficher l’information sur la commande :



>>Créer fichier/répertoire avec un nom déjà existant :





La commamde ‘quitter ‘ pemet de quitter automatiquemet le gestionnaire de fichiers.

Conclusion

La réalisation de ce projet nous a permis de mettre en pratique nos connaissances acquises au cours du structures de données avancées en général et les arbres en particulier .