SI

Rim

S.I

Ouverture du fichier:
FILE *fichier;
fichier = fopen(nom, r);

Fermeture:
int fclose(FILE *fichier);
return 0 si la fermeture s'est bien passée, EOF en cas d'erreur.
ex: FILE *fichier;
fichier = fopen(nom, r);
fclose(fichier);

Destruction:
int remove(char *nom)
return 0 si la fermeture s'est bien passée.
ex: remove(nom);

Renommer:
int rename(char *oldname, char *newname);
return 0 si la fermeture s'est bien passée.

Positionnement du pointeur au début du fichier:
void rewind(FILE *fichier);

Ecriture dans le fich:
int putc(char c, FILE *fichier): écrit la valeur de c à la position courante du pointeur, le pointeur avance d'une case mémoire, retourne EOF en cas d'erreur.
ex: putc('A', fichier);

int putw(int n, FILE *fichier); Idem, n de type int, le pointeur avance du nombre de cases correspondant à la taille d'un int (4 cases). retourne n si l'écriture s'est bien passée.

int fputs(char *chaîne, FILE *fichier); le pointeur avance de la longueur de la chaîne ('\0' n'est pas rangé dans le fich). retourne EOF en cas d'erreur.
ex: fputs(Bonjour!, fichier);

ex: fprintf(fichier, "%s", "rim");
fprintf(fichier, "%d", n);
fprintf(fichier, "%s%d", "rim", n);
le pointeur avance d'autant.

lecture du fichier:
int fgetc(FILE *fichier); lit 1 caractère, mais retourn un int n; retourn EOF si erreur ou fin de fichier, le pointeur avance d'une case.
ex: char c;
c = (char) getc(fichier);

int getw(FILE *fichier);
idem avec un int, le pointeur avance de la taille d'un entier.
ex: int n;
n = getw(fichier);

char *fgets(char *chaine, int n, FILE *fichier);
lit n-1 caractères à partir de la position du pointeur et les range dans une chaine en ajoutant '\0'.

int fread(void *p, int taille bloc, int nb_bloc, FILE *fichier); retourne le nombre de blocs lus et 0 à la fin du fichier.
equivalente à fscanf en lecture
int fscanf(FILE *fichier, char *format, liste d'adresses); equivalente à fprintf en lecture.

Gestion des erreurs:
fopen retourne le pointeur NULL si erreur (impossibilité d'ouvrir le fichier)
fgets retourne le pointeur NULL en cas d'erreur ou si la fin du fichier est atteinte.
int feof(FILE *fichier) retourne 0 tant que la fin du fichier n'est pas atteinte
ferror(FILE *fichier) retourne 1 si une erreur est apparue lors d'une manipulation de fichier, 0 dans le cas contraire.

Fonction particulière aux fichiers à accès direct:
int fseek(FILE *fichier, int offset, int direction); déplace le pointeur de offset cases à partir de direction.
valeurs possibles pour direction:
0 → à partir du début du fichier.
1 → " " de la position courante du p
2 → en arrière à partir de la fin du fichier,
retourne 0 si le pointeur a pu être déplacé.