Makefile

Ce fichier est un "script" de compilation. Il automatise la création du programme exécutable.

SRC = ...: Définit la liste de tous les fichiers de code source (.c) nécessaires pour construire le projet.

NAME = name: Spécifie que le nom du programme final (l'exécutable) sera name.

all: $(NAME): C'est la commande par défaut. Elle indique que pour "tout" construire, il faut créer le fichier name.

$(NAME):: C'est la règle qui explique comment créer name. Elle utilise le compilateur cc avec des options de compilation strictes (-Wextra -Wall -Werror) pour transformer tous les fichiers source en un seul exécutable.

fclean:: Une commande pour nettoyer le répertoire en supprimant le programme exécutable name.

re:: Une commande de raccourci pour re-compiler entièrement le projet. Elle nettoie d'abord (fclean), puis re-construit (all).

rush-02.c

C'est le point d'entrée du programme, là où l'exécution commence.

main: La fonction principale. Son rôle est de gérer les arguments passés au programme lors de son exécution en ligne de commande.

Si le programme est lancé avec un seul argument (ex: ./name 123), il traite cet argument comme le nombre à convertir.

S'il est lancé avec deux arguments (ex: ./name numbers.dict 123), le premier est le dictionnaire à utiliser et le second est le nombre.

Fonctions utilitaires: Il contient également des fonctions de base comme is_space pour vérifier si un caractère est un espace, ou atoi_char qui est une version personnalisée de atoi (conversion d'une chaîne en nombre), bien que son implémentation semble incomplète dans le code fourni.

Traitement du nombre (Conditionnement)

Ces trois fichiers travaillent ensemble pour décomposer le nombre en une série de "jetons" (tokens) numériques que le programme peut ensuite traduire.

number_conditioner.c

Ce fichier découpe le grand nombre en blocs de trois chiffres.

pack_filler: La fonction clé ici. Elle prend un nombre sous forme de chaîne de caractères (ex: "1234567") et le découpe de droite à gauche en "paquets" de trois chiffres. Le résultat serait : {567, 234, 1}.

ft_pack_splitter: La fonction qui effectue la conversion d'un paquet de 1 à 3 caractères en une valeur numérique.

dict_packs_conditioner.c

Ce fichier prend un paquet de trois chiffres (ex: 567) et le décompose en jetons plus simples.

packsofthree_emitter: La fonction principale qui orchestre la décomposition. Par exemple, pour 567, elle va générer les jetons suivants :

Un jeton pour la centaine : 5

Un jeton pour le mot "cent" : 100

Un jeton pour la dizaine : 60

Un jeton pour l'unité : 7

treat_hundreds, treat_tens, treat_units: Fonctions spécialisées pour extraire les centaines, les dizaines et les unités de chaque paquet.

conditioners_bridge.c

Ce fichier sert de pont entre les différentes étapes de conditionnement.

nbr_from_str: La fonction de haut niveau qui pilote le processus. Elle appelle pack_filler pour obtenir les paquets.

ft_transmitter: Itère sur chaque paquet (ex: {567, 234, 1}). Pour chaque paquet, elle appelle packsofthree_emitter pour le décomposer, puis scale_code_emitter.

scale_code_emitter: Ajoute le jeton d'échelle (mille, million, milliard...). Par exemple, après avoir traité le paquet 234, elle ajouterait un jeton pour "mille". Après le paquet 1, elle ajouterait un jeton pour "million". Les échelles sont représentées par des nombres négatifs pour les différencier des nombres simples.

Traduction et Affichage

Ces fichiers gèrent la lecture du dictionnaire et l'affichage du résultat final.

token_reader.c

Ce fichier est le "traducteur". Il recherche la correspondance textuelle pour chaque jeton numérique dans le fichier dictionnaire.

token_reader: La fonction principale. On lui donne un jeton (ex: 100) et le contenu du dictionnaire. Elle parcourt le dictionnaire pour trouver la ligne 100: hundred et retourne le mot "hundred".

match_key_numeric / match_key_scale: Fonctions spécialisées pour trouver soit des nombres simples (comme 20) soit des nombres d'échelle (comme 1000000), qui sont gérés différemment.

printer.c

Ce fichier assemble et affiche la chaîne de caractères finale.

print_tokens: La fonction centrale de ce fichier.

Elle reçoit la liste complète des jetons générés (ex: {1, -2, 2, 100, 30, 4, -1, 5, 100, 60, 7} pour 1,234,567).

Elle lit d'abord le fichier numbers.dict en mémoire.

Ensuite, elle boucle sur chaque jeton, appelle token_reader pour obtenir le mot correspondant, et l'imprime à l'écran, en ajoutant un espace entre chaque mot.

numbers.dict

C'est un fichier de données, pas du code. Il sert de dictionnaire de traduction.

Format: Chaque ligne associe un nombre à son équivalent en toutes lettres (ex: 100: hundred).

Contenu: Il contient les mots pour les chiffres de 0 à 19, les dizaines (20, 30...), le mot "cent", et les grandes échelles (mille, million, milliard, etc.). C'est la source de vérité pour la traduction.

En résumé, le programme découpe le nombre d'entrée en une série de jetons numériques logiques, puis utilise le dictionnaire pour traduire chaque jeton en mot avant d'imprimer la phrase complète.