Pràctica 3: Codificador/descodificador Morse

Programació a Baix Nivell — iTIC

Sebastià Vila-Marta Francisco del-Águila-López

13 d'abril de 2022

Índex

1 Introducció

Aquesta sessió s'organitza com un petit projecte l'objecte del qual és dissenyar i construir una comanda que permeti codificar i descodificar missatges escrits en codi Morse.

En aquesta pràctica es jugarà amb una aplicació construïda amb diversos mòduls i els corresponents tests unitaris. És un petit projecte que posa en valor tots els elements que s'han après fins el moment.

2 Organització

2.1 Lliurament

Lliureu aquesta pràctica seguint les mateixes pautes que s'han emprat a la pràctica 1.

2.2 Mètode de treball

Apliqueu les mateixes pautes que s'han emprat a la pràctica 1.

2.3 Control de versions

Apliqueu les mateixes pautes que s'han emprat a la pràctica 1.

3 L'objectiu

Una vegada finalitzat aquest projecte cada equip ha d'haver creat una nova ordre de la shell que permeti codificar i descodificar missatges entre ASCII i Morse. L'ordre s'anomenarà `morse` i tindrà diverses opcions. El següents retalls de la shell permeten entendre com s'hauria d'usar l'ordre:

```
$ ./morse

HOLA␣MON␣<return>
....␣---␣.-..␣.-␣␣␣--␣---␣-.␣␣␣<return>
```

L'ordre `morse` també es pot usar amb l'opció `-d`, que significa "descodifica", i llavors transforma de Morse a ASCII. Així doncs, usant les pipes de la shell s'hauria de poder fer el següent:

```
$ ./morse | ./morse -d

HOLA␣MON␣<return>
HOLA␣MON␣<return>
```

El format de les dades ASCII i Morse és el d'una seqüencia de caràcters en cada línia.

En el cas del text ASCII, és una seqüència de mots formats per lletres majúscules i dígits. Un mot acaba en un sol espai. El missatge acaba en un LF.

En el cas del text Morse, és una seqüència de mots formats per punts i guions que denoten un caràcter morse. Tot caràcter acaba en un espai i tota paraula acaba en dos espais. El missatge morse acaba amb un LF.

La sintaxi EBNF [Wik20] que correspon a un i altre format és la següent:

```
letter = "A" | "B" | "C" | "D" | "E" | "F" | "G"
       | "H" | "I" | "J" | "K" | "L" | "M" | "N"
       | "O" | "P" | "Q" | "R" | "S" | "T" | "U"
       | "V" | "W" | "X" | "Y" | "Z" ;

digit = "0" | "1" | "2" | "3" | "4" | "5" | "6"
      | "7" | "8" | "9" ;

eol = LF

ascii_char = letter | digit

ascii_word = ascii_char  { ascii_char }  " "

ascii_data = ascii_word  { ascii_word }  eol

morse_symbol = "-" | "."
```

```
morse_char =  morse_symbol  { morse_symbol }  " "

morse_word = morse_char  { morse_char }  "  "

morse_data = morse_word  { morse_word }  eol
```

Es recomanable llegir-se l'apartat 7 per entendre bé el concepte de final de línia (EOL).

TASCA PRÈVIA 1 Llegiu-vos amb cura l'entrada anglesa a la Viquipèdia sobre el codi Morse, [Wik12]. En aquest exercici usarem la codificació ITU. Fixeu-vos especialment en la representació en arbre que se suggereix al final de l'article.

TASCA PRÈVIA 2 Aquest projecte requerirà aplicar tests unitaris. Ja coneixeu aquest mecanisme del món `Python`. En el cas de `C`, però, no hi ha cap eina estàndard per a fer tests. En el nostre cas usarem una llibreria de `C` que s'anomena `CUnits`. Si useu el vostre computador cal que instal·leu els paquets `libcunit1` i `libcunit1-dev` com sempre fent:

```
$ sudo aptitude install libcunit1 libcunit1-dev
```

Fullejeu una mica la documentació, en especial els exemples introductoris, que trobareu a [KS+10; NS07] per tal de conèixer una mica l'eina.

A continuació hi trobareu l'especificació dels diversos mòduls que composen el programa.

4 El mòdul codificador

L'objectiu d'aquest mòdul, de nom `codif`, és codificar i descodificar caràcters d'ASCII a Morse i viceversa. Això es farà mitjançant un objecte privat del mòdul que, d'una forma més o menys sofisticada, emmagatzema una taula de traducció entre ASCII i Morse. EL mòdul disposa de dues classes de recursos: uns permeten crear taules de codificació Morse, els altres permeten codificar caràcters usant una d'aquestes taules. Els tipus i les operacions més importants que ofereix el mòdul són:

morse_char_t	És un tipus de dades que representa una cadena Morse corresponent a un sol caràcter. Com que en el codi ITU la llargada màxima d'un caràcter és de 5 símbols, el representarem com una taula de mida fixa de 6 caràcters. Aquesta taula sempre serà tractada com una cadena de caràcters i, per tant, sempre contindrà el caràcter `'\0'` com a sentinella.
morse_table_t	És un tipus de dades que representa una taula de codificació Morse. Atesa la mida màxima del codi ITU, 5 simbols, i la implementació de la taula que es proposa més endavant, la mida màxima d'aquesta taula és de 2^{5+1} cel·les. Les cel·les són de tipus **char**.
void empty_morse_table(morse_table_t t);	Modifica la taula de codificació t i esborra el seu contingut resultant una taula neta.
void set_translation(morse_table_t t, **char** c, **const** morse_char_t m);	Afegeix a la taula de codificació t la correspondència entre el caràcter c i la seqüència Morse m.
char to_ascii(**const** morse_table_t t, **const** morse_char_t m);	Retorna el caràcter ASCII representat per la sequència Morse m. Si la seqüència Morse no té cap caràcter associat, retorna el caràcter especial `'@'`.
void to_morse(**const** morse_table_t t, **char** c, morse_char_t m);	Modifica m assignant-li la cadena de caràcters que conté la representació Morse del caràcter ASCII c. Si c no conté un caràcter del domini, m és una cadena de longitud zero.

TASCA 3 Creeu el fitxer de header corresponent a aquest mòdul i declareu-hi els tipus i les capçaleres de les funcions públiques del mòdul.

Un exemple típic d'ús del mòdul podria ser el següent:

```
morse_table_t t;

empty_morse_table(t);
set_translation(t, 'F', "..-.");
set_translation(t, 'I', "..");

morse_char_t seq_morse;

to_morse(t, 'F', seq_morse);
printf("%s\n", seq_morse);
```

L'exemple hauría d'escriure pel canal de sortida el text `"..-."`.

Per a implementar aquest mòdul cal decidir primer com s'emmagatzemarà la taula de traducció, és a dir, l'estructura de dades que emmagatzema la correspondència entre ASCII i Morse. A primera vista, podrien usar-se un parell de diccionaris que fessin correspondre ASCII a Morse i Morse a ASCII respectivament. Hi ha dos inconvenients en aquesta possibilitat: l'espai que ocuparia aquesta estructura i el fet que la llibreria estàndard de C no disposa de diccionaris. Optarem doncs per dissenyar una estructura de dades *ad-hoc*.

Noteu que, tal i com ja es deia a [Wik12], la taula de codificació Morse pot ser vista com un arbre binari en que cada node representa un caràcter i les arestes un punt o una ratlla. En aquest

arbre, el camí de l'arrel fins un node correspon amb la codificació Morse del node. Per simplificar assumim d'ara en endavant que l'aresta esquerra d'un node sempre és ratlla i la dreta punt.

En aquest context, saber la lletra corresponent a una seqüencia Morse és molt senzill: només cal anar seguint el camí que indica la seqüència Morse per arribar a un node. Aquest node és la lletra corresponent. De forma simètrica, donat un node que representa una lletra, només cal seguir el camí fins l'arrel per conèixer la seva codificació Morse.

Aquest arbre és notablement equilibrat. Si fos molt desequilibrat, significaria que hi ha símbols Morse molt llargs quan en realitat podríen ser més curts i això és improvable. Per tant és assenyat esperar que aquest arbre sigui raonablement equilibrat.

> TASCA PRÈVIA 4 Repasseu els conceptes relacionats amb els arbres de TECPRO. Dibuixeu un arbre de codificació Morse que contingui tots els símbols de llargades 1 i 2.

La taula de traducció doncs serà un arbre binari. De les diverses formes de representar un arbre binari, usarem la representació compacta en *level-order* sobre una taula. En cas que no recordeu aquesta representació, podeu consultar qualsevol text d'estructures de dades o les referències [Har11; Hol12].

> TASCA PRÈVIA 5 Representeu l'arbre del previ anterior sobre una taula (dibuixada en un paper!). Useu-la per codificar i descodificar un exemple tot i prestant molta atenció en com es consulta la taula en un i altre cas.

TASCA 6 Amb les pistes anteriors implementeu el mòdul `codif.c` usant una representació en arbre sobre una taula. Al mateix temps, creeu un fitxer de nom `test_morse.c` en el que, usant la llibreria `CUnit`, escriureu un test unitari pel mòdul `codif`. Assegureu-vos que el mòdul codif passa els tests correctament.

5 El mòdul itu

La funció d'aquest mòdul és oferir una taula de traducció que implementa la codificació Morse ITU. La taula serà vista com una variable global compartida que defineix aquest mòdul. És molt simple i només ofereix les següents funcionalitats:

`extern morse_table_t itu_table;`	Variable compartida de només lectura que conté una taula de codificació Morse segons ITU. La variable només pot usar-se després d'haver inicialitzat el modul.
`void itu_init(void);`	Inicialitza el mòdul `itu` amb la taula de codificació Morse que correspon a l'estàndard ITU.

TASCA 7 Implementeu el mòdul `itu`. Amplieu el test unitari incloent aquest nou mòdul.

> TASCA PRÈVIA 8 L'ús d'una variable compartida no és l'única manera d'implementar aquest mòdul. En aquest disseny s'ha triat aquesta opció per tal de poder-hi experimentar. Això no obstant, les variables compartides comporten diversos problemes importants. Penseu una mica i feu una llista dels problemes que pot implicar l'ús de variables compartides.

6 El mòdul streamencoder

L'objectiu d'aquest mòdul és afegir una capa sobre `itu` i `codif` que permeti codificar i descodificar tot el contingut d'un stream (canal) usant l'estàndard ITU. Les funcionalitats del mòdul són les següents:

void streamencoder_init(**void**);	Inicialitza el mòdul.
void do_codifica(FILE *in, FILE *out);	Codifica el missatge ASCII del canal in i escriu el resultat Morse en el canal out. in és un stream obert en mode lectura i out un stream obert en mode escriptura. El final de missatge ve donat pel `EOF`.
void do_descodifica(FILE *in, FILE *out);	Descodifica el missatge Morse del canal in i escriu el resultat en el canal out. in és un stream obert en mode lectura i out un stream obert en mode escriptura. El final de missatge ve donat pel `EOF`.

Com veieu, les funcions tenen dos paràmetres de tipus FILE * que representen el canal d'entrada i el de sortida. Una pregunta raonable que us podríeu formular és per quina raó cal fer explícit aquest canal quan en realitat sempre correspondran a stdin i stdout, els canals d'entrada i sortida estàndards. La resposta és que fet d'aquesta forma es facilita molt el test d'aquestes funcions ja que podem fer que llegeixin i escriguin en fitxers específicament dissenyats per a fer el test. És un bon hàbit dissenyar les aplicacions no només pensant en el que han de fer sinó introduint també la facilitat de test com a objectiu.

Tasca 9 Implementeu el mòdul `streamencoder`. Amplieu el test unitari incloent aquest nou mòdul. Per fer el test, tingueu en compte que podeu crear fitxers temporals usant la funció tmpfile(), escriure-hi les dades escaients, i executar una funció de codificació sobre aquest fitxer que deixa el resultat sobre un altre fitxer temporal. Posteriorment podeu determinar si aquest fitxer conté el resultat esperat.

7 El mòdul morse

El mòdul `morse` conté el programa principal. La seva funció és descodificar les opcions que l'usuari ha indicat i operar en conseqüència.

> Tasca prèvia 10 Dibuixeu el graf de dependències entre mòduls que formen el projecte.

Tasca 11 Implementeu el mòdul `morse`. Tingueu en compte que cal fer el tractament escaient dels espais. El mòdul cal que sigui net, polit i ben organitzat!

Apèndix: sobre el final de línia

El concepte de «final de línia» sempre és una mica embolicat. La confusió té l'orígen en qüestions conceptuals i històriques. A continuació mirem d'aclarir-ho.

1. Els caràcters ASCII. La taula de codificació de caràcters ASCII té diversos caràcters que estan relacionats amb el concepte de línia. Per entendre'ls cal recordar que la taula ASCII

és antiga i es remunta al temps dels teletips, una mena d'impressores rudimentàries similars a les màquines d'escriure mecàniques.

Els caràcters ASCII són en realitat ordres per a ser enviades a un teletip. Així, si s'envia l'ordre `A`, el teletip imprimeix en el paper la lletra a; si s'envia la lletra `+`, imprimeix en el paper el signe de sumar. Això explica l'existència de «caràcters estranys» a la taula ASCII. Prenen sentit quan s'entenen com a ordres per a un teletip. Per exemple, si s'envia el caràcter de codi `0x07` el teletip fa sonar una campaneta: Ding!, per aquesta raó aquest ASCII caràcter s'anomena BEL(L).

Hi ha certes ordres —caràcters ASCII— interessants de conèixer. L'ordre `0x0a`, coneguda com a LF, provocava un `line feed` en el teletip. És a dir feia avançar el paper una línia. L'ordre `0x0d`, coneguda com CR, provocava un *carriage return*. És a dir recol·locava el capçal d'impressió a la columna 1 del paper.

2. El format dels fitxers de text. Un fitxer de text eś una seqüència de caràcters però sovint interesa veure'l organitzat com una seqüència de línies. A tal efecte cal determinar quin caràcter pot fer la funció de sentinella d'una línia, i.e.: de «final de línia». Noteu que ha de ser un caràcter obligatòriament, atès que un fitxer de text no conté altra cosa.

   Tradicionalment s'han usat els caràcters ASCII LF i/o CR per a aquesta funció. Van bé per que no entren amb conflicte amb les lletres, xifres i altres caràcters ordinaris. La qüestió és que *no hi ha hagut un consens ben establert en quin dels caràcters s'ha d'usar*. Hi ha sistemes operatius que usen LF, altres usen CR i encara altres usen ambdós CR+LF. El món UNIX usa LF. El món Windows usa CR+LF. Per aquesta i només per aquesta raó els fitxers de text en un i altre sistema són lleugerament diferents. La diferència és tant insignificant que moltes eines poden treballar en un i altre format indistintament. `Emacs`, per exemple, treballa correctament en ambdós formats i pot convertir d'una format a l'altre.

3. El teclat. Un teclat és un dispositiu d'entrada amb tecles. Quan es prem una tecla s'envia un senyal al computador i aquest, amb la mediació del sistema operatiu, l'interpreta d'una certa manera. Una tecla habitual és la tecla «return». La interpretació d'aquesta tecla depen del software però moltes vegades s'usa per indicar «final de línia». Noteu que, en general, no és el mateix «final de línia» que el caràcter ASCII CR (*carriage return*).

4. En el llenguatge `C` els caràcters es denoten envoltant-los de cometes simples. Quan el caràcter no és imprimible —no té un gliph associat— el caràcter s'escriu amnb una seqüència precedida d'una barra. Així, per exemple, el caràcter BELL s'escriu com `'\a'`; el LF com `'\n'` i el CR com `'\r'`.

   El llenguatge `C` està definit de forma que fa abstracció de com es representa el «final de línia». Quan llegint un fitxer de text es troba un «final de línia», `C` sempre torna `'\n'`, independentment del sistema operatiu en que es treballa. Simètricament, quan s'escriu en un fitxer de text un `'\n'`, en el fitxer s'hi desa un LF o un CR+LF segons sigui el conveni del sistema operatiu usat. Així, en un programa `C` el «final de línia» sempre és el `'\n'`.

   Per contra, si el fitxer s'obre en mode binari, aleshores es veu la codificació física del «final de línia», que canvia segons el sistema.

Referències

[Har11] Douglas Wilhelm Harder. *Complete Binary Trees. Algorithms and Data Structures Course Notes*. Ang. Dept. of Electrical i Computer Engineering, University of Waterloo. 2011. URL: `https://ece.uwaterloo.ca/~cmoreno/ece250/4.06.CompleteBinaryTrees.pdf` (cons. 20-02-2017).

[Hol12] Robert C. Holte. *Implementing a Tree in an Array. Data Structures Course Lecture Notes*. Ang. University of Ottawa. 2012. URL: `http://webdocs.cs.ualberta.ca/~holte/T26/tree-as-array.html` (cons. 20-02-2017).

[KS+10] Anil Kumar, Jerry St.Clair et al. *CUnit. A Unit Testing Framework for C*. Ang. Dept. of Electrical and Computer Engineering, University of Waterloo. 2010. URL: `http://cunit.sourceforge.net` (cons. 20-02-2017).

[NS07] Brian Nielsen i Arne Skou. *Introduction to C Unit Testing (CUnit)*. Ang. Slides. Test and Verification 2007 course notes. School of Information and Communication Technology, Aalborg University. Denmark, 2007. 29 pàg. URL: `http://www.cs.aau.dk/~bnielsen/TOV07/lektioner/cunit-intro-07.pdf` (cons. 20-02-2017).

[Wik12] Wikipedia contributors. *Morse Code*. Ang. Wikipedia, The Free Encyclopedia. 2012. URL: `http://en.wikipedia.org/wiki/Morse_code` (cons. 20-02-2017).

[Wik20] Wikipedia contributors. *Extended Backus-Naur form*. Ang. Wikipedia, The Free Encyclopedia. 2020. URL: `https://en.wikipedia.org/wiki/Extended_Backus%E2%80%93Naur_form` (cons. 17-03-2020).