Pràctica 2: Programes menys senzills en C

Programació a Baix Nivell — iTIC

Sebastià Vila-Marta

20 de febrer de 2017

Índex

1 Organització

Aquesta sessió s'organitza com una seqüencia de problemes de dificultat creixent que van entrenant en l'ús del llenguatge C. La idea és anar-los resolent, implementant i provant un darrera l'altre fins on sigui possible. Us recomanem que en el vostre temps d'estudi els acabeu de resoldre tots. L'objectiu final és anar aconseguint agilitat amb la sintaxi i les eines de treball relacionades amb el llenguatge C. En aquest cas l'èmfasi està centrat en l'ús de tipus de dades, apuntadors i funcions. En aquesta pràctica el computador que actuarà de target serà la pròpia estació de treball i el sistema operatiu serà GNU/Linux.

1.1 Lliurables

Aquesta pràctica no comporta lliurables atès que el domini del llenguatge i les eines són fonamentals per a la resta del curs i resten suficientment avaluades.

2 Exercicis

Exercici 2.1 Implementeu dues funcions per a la entrada i sortida de bytes. La primera, `read_byte()` ha de llegir un byte en format hexadecimal del canal d'entrada i tornar un valor de tipus `uint8_t`. La segona, `write_byte(uint8_t b)`, ha d'escriure en hexadecimal el byte `b` en el canal de sortida.

Afegiu un programa principal que us permeti fer test d'aquestes funcions.

Exercici 2.2 Dissenyeu i implementeu un programa que llegeix del canal d'entrada una seqüència de 20 enters i calcula quin enter apareix més vegades en la seqüència.

Exercici 2.3 Dissenyeu i implementeu una ordre per convertir bytes del format ASCII a binari. L'ordre llegeix pel canal d'entrada una seqüència de Bytes sense signe en format hexadecimal fins que arriba el Byte 0. Pel canal de sortida va escrivint la mateixa seqüència en format binari (i per tant generalment no imprimible).

Per comprovar el seu funcionament, emmagatzemeu el resultat d'executar l'ordre en un fitxer i editeu-lo usant el mode `hexl` d'`emacs`.

Exercici 2.4 Dissenyeu una funció amb prototip:

void capgira(**int** n, **char** t[n]);

Que capgiri el contingut de la taula `t`.

Exercici 2.5 Dissenyeu i implementeu una cua usant una taula circular. La idea és implementar un mòdul de `C`, format pels fitxers `cua.c` i `cua.h` que defineixin les funcions següents:

cua_t	És un tipus de dades que representa una cua circular d'elements de tipus **char** i de mida MAXCUA, on MAXCUA és una constant que val 20.
bool cua_plena(**const** cua_t ∗**const** c)	Retorna true si la cua c és plena.
bool cua_buida(**const** cua_t ∗**const** c)	Retorna true si la cua c és buida.
int cua_get(cua_t ∗**const** c)	Desencua un element de c i el retorna. Si la cua és buida torna −1. Aquesta és la raó per la qual torna un **int** en comptes d'un **char**. Si el valor retornat és positiu, la seva conversió a caràcter resulta en l'element desencuat. Modifica c per efecte lateral.
void cua_put(cua_t ∗**const** c, **char** e)	Encua l'element e a la cua c, que es modifica per efecte lateral. Si la cua és plena, no s'encua res.

A tal efecte, seguiu els següents passos:

1. Investigueu com s'implementa una cua circular. Podeu usar qualsevol referència clàssica de l'àmbit de les estructures de dades.
2. Creeu un header `cua.h` en què hi afegireu la definició de la constant MAXCUA, el tipus de dades `cua_t` i els prototips de les seves operacions.
3. Creeu una implementació `cua.c` en què implementareu les operacions. Aquest fitxer inclou `cua.h`.
4. Finalment escriviu un programa que permet comprovar el funcionament de la cua. Opcionalment, podeu fer les proves usant un entorn de test com ara `CUnits`, que usarem a la pràctica 3 i que us ofereix unes funcionalitats similars a les del `doctest` a `Python`.