Parte 1 dell’anno: Parte Hardware

Parte 2 dell’anno: Sistemi Operativi

Il C è un linguaggio che è stato sviluppato principalmente per la realizzazione di sistema operativi. È stato creato da un gruppo di “hackers” per progettare il sistema operativo Unix.

Su un sistema Unix “man” è la funzione che restituisce il manuale per la programmazione del/sul (?) sistema operativo.

Il Punto di partenza è la codifica delle informazioni sul sistema di calcolo.

La rappresentazione delle informazioni è ricondotta a una codifica di tipo numerico. Ogni informazione può infatti venir convertita in forma numerica e caricata su un calcolatore costruito a tale scopo.

Quando si scrive un numero su un foglio di carta si sta assumendo che il lettore conosca il codice che si sta usando (quella decimale dei numeri). Tra le caratteristiche di queste rappresentazioni vi sta innanzitutto che sono di lunghezza variabile: 2 è un numero piccolo, 27 è un numero un po' più grande e usa un numero maggiore di cifre, tuttavia la forma in cui sono scritti è sempre minimale (evitando gli zeri inutili in 00002). Quando si passa a un sistema di calcolo invece il codice è a LUNGHEZZA FISSA (Il numero di cifre da usare è prefissato). Questo impone delle limitazioni su quello che possiamo scrivere (in caso si usino 4 cifre si potrà scrivere al massimo 9999) inoltre la scrittura non è più minimale (2 diventerebbe, in caso si usino 4 cifre, 0002).

La verità è che anche quando si scrivono numeri su un foglio di carta esiste un limite fisico (la dimensione del foglio di carta), quindi anche quando si parla di hardware bisogna mettere dei vincoli sul massimo (nonché fisso) numero di cifre in modo da ottimizzare l’attribuzione di spazi di memoria. (I limiti derivano dall’esterno quindi, dall’Hardware…).

La codifica decimale inoltre fa sì che a ogni cifra sia associato un coefficiente moltiplicativo che prosegue in potenze di 10 (unità, DECINE, CENTINAIA, MIGLIAIA, indicano i coefficienti moltiplicativi -> 1, 10, 100, 1000). Una cosa analoga sì può avere con le basi maggiori di 10 (basta inventarsi simboli che rappresentino il 10, il’11, 12 ecc. -> si sceglie per convenzione le lettere; quindi in caso di base esadecimale i numeri dal 10 al 15 sono rappresentati dalle lettere dalla A alla F).

Il vantaggio sostanziale della base 2 è quello di essere facilmente realizzabile perché per realizzarla in un calcolatore è sufficiente distinguere tra solo due stati di una variabile fisica (come una differenza di potenziale). (Noi quando proviamo a fare a mano dei procedimenti che vengono fatti in automatico dai sistemi di calcolo usiamo la base 8 o la base 16 per comodità (?)).

I vincoli sul massimo numero rappresentabile nei calcolatori a codifica binaria sono, per gli interi, (2^16 -1) --> 16 bit. Si può aumentare tramite una rappresentazione a 32 bit o 64 bit (che a questo punto rappresenterebbero dei long?). Per rappresentare cose diverse dai numeri interi; quindi, per esempio, i colori, le lettere, ecc. Si elencano tali elementi e si sfrutta la loro posizione Numerica nell’elenco per poterli codificare numericamente. I caratteri, per esempio, utilizzando una rappresentazione a 7 bit, che permette la rappresentazione di fino a 128 caratteri diversi (-> ASCII).

Per convertire altre forme di input, quali output di un microfono, si utilizzano dei CAMPIONI, che vengono presi a diversi istanti molto vicini e di cui si osserva qualche condizione, come una diff. di potenziale, che viene memorizzata in una sequenza / elenco numerico per poter essere riprodotta.

Analogamente si sfrutta lo stesso procedimento con le Immagini, si separa l’immagine in pixel, a cui viene associato un numero (che corrisponde alla codifica rgb a 24bit, o rgba a 32bit) e che viene memorizzato per permetterne la rappresentazione in un secondo momento.

Per codificare i numeri con segno (-1,-2…) si usa un bit che codifica l’esponente di un -1. Quando l’esponente è pari il -1 diventa positivo e quindi il numero (o meglio il Modulo) con cui viene moltiplicato diventa positivo, quando è dispari il -1 resta -1 e il numero con cui viene moltiplicato dà un prodotto negativo. Questo bit che codifica per il segno si trova, nella codifica a 8 bit per esempio, nella posizione più a sinistra. Ciò fa sì che +0 e -0 possono essere scritti come due valori diversi. Questo è uno svantaggio dal punto di vista informatico, perché essendoci due (o più) modi di rappresentare lo stesso valore la rappresentazione si dice “non canonica”. Ciò complica i confronti -> Se io voglio chiedermi se un numero è uguale a un altro, se la rappresentazione è canonica basta confrontare le stringhe di bit dei due numeri e vedere se corrispondono. Ma se la rappresentazione è non canonica ciò non può avvenire. Una possibile soluzione potrebbe essere la “rappresentazione in complemento a 1” -> Quando l’ottavo bit diventa 1 il valore degli altri bit viene invertito nella rappresentazione del numero, che a questo punto viene considerato di segno negativo (quindi ad esempio 2 che è 00000010 col segno - davanti diventa 11111101). Tuttavia, nel complemento a 1 lo zero rimane rappresentabile sia come 00000000 che come 11111111.

La rappresentazione in complemento a 2, che è la più utilizzata oggi, risolve invece questo problema. Se all’ottavo bit è presente un 1 anziché uno zero, il numero diventa negativo poiché al numero positivo, costituito dai bit che vanno dal primo al settimo, viene sottratto 128. Ecco quindi che l’unico modo per rappresentare lo zero è con 00000000 (mentre per esempio -1 è 11111111 poiché questo valore indica 127-128). Bisogna puntualizzare che tale complemento permette di rappresentare il numero -128 (come 10000000) ma non +128, in quanto i numeri positivi si fermano a 127.