

Carlo A. Mazzone

La shell testuale

Utenti avanzati con la riga di comando

CODICE SORGENTE

"Quanto più grande il potere, tanto più pericoloso l'abuso."

Edmund Burke

"Quanto a me, sono un principe libero e ho tanta autorità per fare guerra al mondo intero, come se disponessi di cento vascelli sul mare o centomila uomini sulla terraferma, ecco ciò che sento."

Samuel Bellamy

Capitolo 1 – Software, sistemi operativi e reti di computer

Propongo di considerare questa domanda: "Le macchine sono in grado di pensare?"

Alan Turing

0100 0000 0000 1001

READ 9

Capitolo 2 – CMD.EXE: il DOS è vivo

Una generazione che ignora la storia non ha passato... né futuro.

Robert Anson Heinlein

Principali comandi:

ASSOC	Visualizza o modifica le associazioni alle estensioni dei file.
ATTRIB	Visualizza o modifica gli attributi del file.
BREAK	Attiva o disattiva il controllo esteso di CTRL+C.
BCDEDIT	Imposta le proprietà nel database di avvio per il controllo del caricamento avvio.
CACLS	Visualizza o modifica gli elenchi di controllo di accesso (ACL) dei file.
CALL	Richiama un programma batch da un altro.
CD	Visualizza il nome della directory corrente o consente di passare a un'altra directory.
CHCP	Visualizza o imposta il numero di tabella codici attiva.
CHDIR	Visualizza il nome della directory corrente o consente di passare a un'altra directory.
CHKDSK	Controlla un disco e visualizza il relativo rapporto sullo stato.
CHKNTFS	Visualizza o modifica la verifica di un disco durante l'avvio.
CLS	Cancella lo schermo.
CMD	Avvia una nuova istanza dell'interprete dei comandi di Windows.
COLOR	Imposta i colori predefiniti in primo piano e dello sfondo della console.
COMP	Confronta il contenuto di due file o di due gruppi di file.
COMPACT	Visualizza o modifica la compressione di file su partizioni NTFS.
CONVERT	Converte volumi FAT in NTFS. Non è possibile convertire l'unità in uso.
COPY	Copia uno o più file in un'altra posizione.
DATE	Visualizza o imposta la data.
DEL	Elimina uno o più file.
DIR	Visualizza un elenco di file e sottodirectory in una directory.
DISKPART	Visualizza o configura le proprietà di Partizione disco.

DOSKEY	Modifica righe di comando, richiama comandi di Windows e crea macro.
DRIVERQUERY	Visualizza stato e proprietà del driver di dispositivo corrente.
ECHO	Visualizza messaggi o attiva e disattiva la ripetizione a video dei comandi.
ENDLOCAL	Termina la localizzazione di modifiche di ambiente in un file batch.
ERASE	Elimina uno o più file.
EXIT	Termina il programma CMD.EXE (interprete dei comandi).
FC	Confronta due file o set di file e ne visualizza le differenze.
FIND	Ricerca una stringa di testo in uno o più file.
FINDSTR	Ricerca stringhe nei file.
FOR	Esegue un comando specificato per ogni file in un set di file.
FORMAT	Formatta un disco per l'utilizzo con Windows.
FSUTIL	Visualizza o configura le proprietà del file system.
FTYPE	Visualizza o modifica i tipi di file utilizzati nelle associazioni delle estensioni di file.
GOTO	Indirizza l'interprete dei comandi di Windows a una riga con etichetta in un programma batch.
GPRESULT	Visualizza le informazioni relative a Criteri di gruppo per il computer o l'utente.
GRAFTABL	Abilita Windows alla visualizzazione di un set di caratteri estesi in modalità grafica.
HELP	Fornisce informazioni della Guida per i comandi di Windows.
ICACLS	Visualizza, modifica ed esegue il backup o il ripristino degli ACL per file e directory.
IF	Esegue un'elaborazione condizionale in un programma batch.
LABEL	Crea, cambia o elimina l'etichetta di volume di un disco.
MD	Crea una directory.
MKDIR	Crea una directory.
MKLINK	Crea collegamenti simbolici e reali
MODE	Configura un dispositivo di sistema.
MORE	Visualizza l'output una schermata alla volta.
MOVE	Sposta uno o più file da una directory a un'altra directory.
OPENFILES	Visualizza i file aperti dagli utenti remoti per una determinata condivisione di file.
PATH	Visualizza o imposta un percorso di ricerca per file eseguibili.
PAUSE	Sospende l'elaborazione di un file batch e visualizza un messaggio.
POPD	Ripristina il valore precedente della directory corrente salvato con PUSHHD.
PRINT	Stampa un file di testo.
PROMPT	Cambia il prompt dei comandi di Windows.
PUSHHD	Salva la directory corrente e poi la cambia.
RD	Rimuove una directory.
RECOVER	Recupera le informazioni leggibili da un disco danneggiato o difettoso.
REM	Registra commenti (note) in file batch o CONFIG.SYS.
REN	Rinomina uno o più file.
RENAME	Rinomina uno o più file.
REPLACE	Sostituisce i file.
RMDIR	Rimuove una directory.
ROBOCOPY	Utilità avanzata per la copia di file e alberi di directory
SET	Visualizza, imposta o rimuove variabili di ambiente di Windows.
SETLOCAL	Inizia la localizzazione di modifiche di ambiente in un file batch.
SC	Visualizza o configura i servizi (processi in background).
SCHTASKS	Pianifica comandi e programmi da eseguire su un determinato computer.
SHIFT	Modifica la posizione di parametri sostituibili in file batch.
SHUTDOWN	Consente il corretto arresto del computer in modalità locale e remota.
SORT	Ordina l'input.
START	Avvia una finestra separata per l'esecuzione del programma o comando specificato.
SUBST	Associa il percorso a una lettera di unità.
SYSTEMINFO	Visualizza la configurazione e le proprietà specifiche del computer.
TASKLIST	Visualizza tutte le attività in esecuzione inclusi i servizi.
TASKKILL	Interrompe o arresta un processo o un'applicazione in esecuzione.
TIME	Visualizza o imposta l'ora del sistema.
TITLE	Imposta il titolo della finestra per una sessione CMD.EXE.
TREE	Visualizza graficamente la struttura di directory di un'unità o percorso.
TYPE	Visualizza il contenuto di un file di testo.
VER	Visualizza la versione di Windows.
VERIFY	Indica se effettuare o meno la verifica della corretta scrittura

	dei file sul disco.
VOL	Visualizza l'etichetta di volume e il numero di serie del disco.
XCOPY	Copia file e alberi di directory.
WMIC	Visualizza le informazioni relative a WMI all'interno della shell dei comandi interattivi.

Capitolo 3 – File batch e scripting sotto Windows

Si possono ripetere le azioni: i pensieri non ritornano mai identici.

Francesco Burdin

```
@ECHO OFF
CLS
SET /P pippo=Inserisci un valore:
ECHO Hai inserito %pippo%

-----

@ECHO OFF
:menu
CLS
ECHO *****
ECHO *           VERIFICA RETE
ECHO *
ECHO *   1. VISUALIZZA IMPOSTAZIONI DI RETE
ECHO *   2. TEST CONNESSIONE RETE LOCALE
ECHO *   3. TEST CONNESSIONE INTERNET
ECHO *
ECHO *   Q. Esce e torna al prompt dei comandi
ECHO *****

SET GW=192.168.1.1
SET /P scelta=SCEGLI:

IF %scelta%==1 GOTO ipconfig
IF %scelta%==2 GOTO gateway
IF %scelta%==3 GOTO internet

IF %scelta%==Q GOTO :EOF
IF %scelta%==q GOTO :EOF

:ipconfig
ECHO Rileviamo le configurazioni delle schede di rete
IPCONFIG /ALL
PAUSE
GOTO menu

:gateway
ECHO Eseguiamo un ping al default gateway
PING %GW%
PAUSE
GOTO menu

:internet
ECHO Test connessione Internet
PING 8.8.8.8
PAUSE
GOTO menu

-----
```

```
@ECHO OFF
CLS
SET /P nomefile=Inserisci il nome del file da eliminare:
DEL "%nomefile%"
```

```
-----

@ECHO OFF
CLS
ECHO Nome programma: %0
ECHO Primo argomento: %1
ECHO Secondo argomento: %2
```

```
-----

@ECHO OFF
SET COUNT=0
FOR %%X IN (*) DO SET /A COUNT+=1
ECHO Numero di argomenti sulla riga di comando: %COUNT%
```

```
-----

@ECHO OFF

REM *****
REM *** Clear Print Queue ***
REM *** Carlo A. Mazzone ***
REM *****

CLS
ECHO CANCELLAZIONE CODA DI STAMPA

REM Stoppiamo il servizio spooler di stampa
NET stop spooler

REM Cancelliamo i file dalla coda di stampa
DEL /F /S /Q %systemroot%\System32\spool\PRINTERS\*

REM Riavvio del servizio spooler di stampa
NET start spooler
```

```
-----

@ECHO OFF

CLS
ECHO *****
ECHO *** BACKUP UTILITY ***
ECHO *** Carlo A. Mazzone ***
ECHO *****

SET LOGFILE=c:\BackupLogs.txt
SET REMOTEDISK=X:

ECHO ***** >> %LOGFILE%

DATE /T >> %LOGFILE%
TIME /T >> %LOGFILE%

ECHO Inizio operazioni ...
ECHO Inizio operazioni >> %LOGFILE%

REM Test esistenza directory destinazione
```

```

IF not EXIST %REMOTEDISK% GOTO notarget

ECHO Copia CARTELLA_1 in corso ...
ECHO COPIA CARTELLA_1 >> %LOGFILE%
xcopy "C:\CARTELLA_1\*.*)" %REMOTEDISK%\DATI_BACKUP\ /E /D /Y >> %LOGFILE%

ECHO Copia CARTELLA_2 in corso ...
ECHO COPIA CARTELLA_2 >> %LOGFILE%
xcopy "C:\CARTELLA_2\*.*)" %REMOTEDISK%\DATI_BACKUP\ /E /D /Y >> %LOGFILE%

GOTO fine

:notarget
ECHO Errore - target NON disponibile >> %LOGFILE%
GOTO fine

:fine
DATE /T >> %LOGFILE%
TIME /T >> %LOGFILE%

ECHO Fine operazioni >> %LOGFILE%

ECHO ***** >> %LOGFILE%

```

Capitolo 4 – Comandi Windows per la diagnostica di rete

La parola **hacking** evoca alcuni **stereotipi**: vandalismo elettronico, spionaggio industriale e **personaggi pittoreschi** con capigliature variopinte e anelli nel naso. È molto diffuso il **pregiudizio** secondo cui l'**hacker** sarebbe un **criminale**. Benché vi siano senz'altro **individui** che utilizzano le **tecniche di hacking** per scopi illeciti, l'hacking è tutt'altra cosa e tende semmai ad **assecondare il rispetto delle leggi** anziché la loro violazione.

Jon Erickson

```

nslookup www.tesseract.it

nslookup -type=soa www.tesseract.it

nslookup www.tesseract.it ns1.register.it

tracert 8.8.8.8

```

Capitolo 5 – Una PowerShell per il power user

Se sei **forte**, sii **mite** e **pacifico**, in modo che chi ti sta vicino abbia **rispetto** di te più che **paura**.

Chilone di Sparta

```
#Esempio di costrutto selezione
$x = 11
if ($x -gt 10)
{
    Write-Host "Il valore $x è maggiore di 10."
}

if ($x -gt 10)
{
    Write-Host "Il valore $x è maggiore di 10."
}

#Esempio di costrutto selezione
Clear-Host
Write-Host "ESEMPIO DI UTILIZZO DEL COSTRUTTO SELEZIONE."
$x = 9
if ($x -gt 10)
{
    Write-Host "Il valore $x è maggiore di 10."
}
elseif ($x -lt 10)
{
    Write-Host "Il valore $x è minore di 10."
}
else
{
    Write-Host "Il valore $x è uguale a 10."
}

#Esempio di costrutto Switch
Clear-Host
Write-Host "Che giorno è?"
$oggi = 2
switch ($oggi)
{
    1 { $giorno = 'lunedì' }
    2 { $giorno = 'martedì' }
    3 { $giorno = 'mercoledì' }
    4 { $giorno = 'giovedì' }
    5 { $giorno = 'venerdì' }
    6 { $giorno = 'sabato' }
    7 { $giorno = 'domenica' }
}

$giorno

for($i=1; $i -le 5; $i++)
{
    Write-Host "Salve mondo";
}
```

```

$i=1;
for( $i -le 5; $i++)
{
    Write-Host "Salve mondo";
}

for($i=1; $i -le 5; $i = $i+1)
{
    Write-Host "Salve mondo";
}

for($i=1; $i -le 5; $i++)
{
    Write-Host "Salve mondo";
}
Write-Host $i;
for($i=1; $i -le 10; $i=$i+2)
{
    Write-Host $i;
}

for($i=2; $i -le 10; $i=$i+2)
{
    Write-Host $i;
}

$i=1;
while($i -le 5)
{
    Write-Host "Qui si preferisce il while";
    $i++;
}

$potenze_di_due = 1,2,4,8,16,32,64,128
$i=0;
while($i -le 7)
{
    $potenze_di_due[$i];
    $i++;
}

$i=1;
do
{
    Write-Host "Perché noi siamo controcorrente";
    $i++;
}
while($i -le 5);

while($i -le 5) { Write-Host "Qui si preferisce il while"; $i++; }

$i=1;
do
{
    Write-Host "Ancora al contrario";
    $i++;
}

```

```

until($i -gt 5);

$i=1;
while($i -le 5)
{
    Write-Host "Conto fino a cinque";
    $i++;
}

$i=1;
while($i -le 5)
{
    Write-Host "Fermate il mondo, voglio scendere!";
}

$i=1;
while($true)
{
    Write-Host "Ho contato fino a $i";
    $i++;

    if($i -eq 101)
    {
        break;
    }
}

for(;;)
{
    Write-Host "Se non digiti CTRL+C non mi fermo!";
}

$nome = "C","a","r","l","o"
foreach ($lettera in $nome)
{
    Write-Host $lettera
}

#impostiamo come cartella corrente quella di installazione di Windows
Set-Location $env:SystemRoot

foreach ($file in Get-ChildItem)
{
    Write-Host $file;
}

foreach ($file in Get-ChildItem)
{
    Write-Host $file.Name `t $file.Length;
}

foreach ($file in Get-ChildItem)
{
    if ($file.length -gt 1)
    {
        Write-Host $file.Name `t $file.Length;
    }
}

```

```

function Saluto
{
    Write-Host "Saluti da una funzione!";
}

Saluto;

function Saluto
{
    Param($messaggio)
    Write-Host $messaggio;
}

function Saluto ($messaggio)
{
    Write-Host $messaggio;
}

Saluto -messaggio "Buon Non Compleanno a tutti!";

function Saluto ($messaggio = "Buon divertimento con PowerShell")
{
    Write-Host $messaggio;
}
Saluto;
Saluto -messaggio "Buon Non Compleanno a tutti!";

function Math-Minus ($a, $b)
{
    $result = $a - $b;
    Write-Host $result;
}

function Math-Minus ([int]$a, [int]$b)
{
    $result = $a - $b;
    Write-Host $result;
}

function Get-Product ($a, $b)
{
    $result = $a*$b;
    Write-Host $result;
}

function Get-Division ($a, $b)
{
    $result = $a/$b;
    Write-Host $result;
}

```

Capitolo 6 – Finestre e .NET

La mia **casa** è piccola ma le sue **finestre** si aprono su un **mondo infinito**.

```

$myForm = New-Object system.Windows.Forms.Form;
$myForm.Text = "Applicazione di esempio per PowerShell";
$myForm.ShowDialog();

$myIcon = New-Object system.drawing.icon("$PSScriptRoot\mondo.ico");
$myForm.Icon = $myIcon;

$OKButton = New-Object System.Windows.Forms.Button;
$OKButton.Text = 'OK';
$myForm.Controls.Add($OKButton);

$OKButton.Location = New-Object System.Drawing.Point(100,180)

Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Drawing

#Creo un nuovo oggetto di tipo Form
$myForm = New-Object system.Windows.Forms.Form;

#Imposto title bar del form
$myForm.Text = "Applicazione di esempio per PowerShell";

#Associo icona
$myIcon = New-Object system.drawing.icon("$PSScriptRoot\mondo.ico");
$myForm.Icon = $myIcon;

#Imposto le dimensioni form
$myForm.Size = New-Object System.Drawing.Size(400,270);

#Setting posizione iniziale
$myForm.StartPosition = 'CenterScreen';

#Forzo la finestra ad apparire al di sopra di qualsiasi altra finestra
$myForm.Topmost = $true;

#Gestione bottone
$OKButton = New-Object System.Windows.Forms.Button;
$OKButton.Location = New-Object System.Drawing.Point(100,180)
$OKButton.Size = New-Object System.Drawing.Size(80,25)
$OKButton.Text = 'OK';
$myForm.Controls.Add($OKButton);

#Visualizzo form
$myForm.ShowDialog();

#-----
# Applicazione GUI di esempio con PowerShell
# Autore: Carlo A. Mazzone
#-----

Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Drawing

#Creo un nuovo oggetto di tipo Form
$myForm = New-Object system.Windows.Forms.Form;

#Imposto title bar del form
$myForm.Text = "Applicazione di esempio per PowerShell";

#Associo icona

```

```

$myIcon = New-Object system.drawing.icon("$PSScriptRoot\mondo.ico");
$myForm.Icon = $myIcon;

#Imposto le dimensioni form
$myForm.Size = New-Object System.Drawing.Size(400,270);

#Setting posizione iniziale
$myForm.StartPosition = 'CenterScreen';

#Forzo la finestra ad apparire al di sopra di qualsiasi altra finestra
$myForm.Topmost = $true;

#Gestione bottone OK
$OKButton = New-Object System.Windows.Forms.Button;
$OKButton.Location = New-Object System.Drawing.Point(100,180)
$OKButton.Size = New-Object System.Drawing.Size(80,25)
$OKButton.Text = 'OK';

$myForm.Controls.Add($OKButton);

#Gestione bottone Chiudi
$CloseButton = New-Object System.Windows.Forms.Button;
$CloseButton.Location = New-Object System.Drawing.Point(200,180)
$CloseButton.Size = New-Object System.Drawing.Size(80,25)
$CloseButton.Text = 'Chiudi';

$myForm.Controls.Add($CloseButton);

#Event handler bottone OK
$OKButton.Add_Click(
    {
        [System.Windows.Forms.MessageBox]::Show("Salve mondo da una GUI PowerShell");
    }
)

#Event handler bottone Chiudi
$CloseButton.Add_Click(
    {
        $myForm.Close();
    }
)

#Visualizzo form
$myForm.ShowDialog();

```

Capitolo 7 – Un pinguino per amico: il sistema operativo Linux

La ragione per cui ho cominciato a scrivere degli hacker, quei programmatori e progettisti di computer che considerano l'informatica come la cosa più importante al mondo, era che si trattava di gente affascinante.

Steven Levy

```
wslconfig /list
```

```
wslconfig /setdefault kali-linux
```

Capitolo 8 – Linux su Android

Verrà un **giorno**, e non è molto lontano, in cui **potremo** concludere affari, studiare, conoscere il mondo e le sue culture, assistere a importanti spettacoli, stringere amicizie, visitare i negozi del quartiere e mostrare fotografie a parenti lontani, tutto senza muoverci dalla scrivania o dalla poltrona.

Bill Gates

```
apt-get update
```

```
apt-get upgrade
```

Capitolo 9 – Linux e i comandi testuali

L'unico **tiranno** che accetto in questo mondo è la voce silenziosa dentro di me.

Mahatma Gandhi

```
apt-get upgrade nomePacchetto
```

```
apt-get install nomePacchetto
```

```
apt-get remove nomePacchetto
```

```
apt-cache show nomePacchetto
```

Capitolo 10 – Programmazione nella Bash

La maggior parte dei **programmatici bravi** non programma perché si aspetta di essere pagato o essere adulato da parte del pubblico, ma perché

```
# Un primo semplice esempio
echo Salve mondo dalla shell
```

```
#!/bin/bash
# Test variabili
saluto="Ciao ciao dal Terminale"
echo $saluto
```

```
#!/bin/bash
nome="Carlo"
saluto="Ciao da $nome. Come state?"
echo $saluto
```

```
#!/bin/bash
echo "Benvenuto, dimmi il tuo nome: "
read nome
saluto="Ciao $nome, io sono il Terminale."
echo $saluto
```

```
#!/bin/bash
let a=3
let b=2
let c=$a+$b
echo $c
```

```
#!/bin/bash
let a=3
let b=2
echo $a+$b
```

```
#!/bin/bash
echo "Ciao, io sono $0. Di seguito i parametri che mi hai passato."
echo "Primo argomento: $1"
echo "Secondo argomento: $2"
echo "Terzo argomento: $3"
```

```
#!/bin/bash
echo "Hai indicato $# argomenti."
echo "Di seguito te li elenco:"
echo $*
```

```
#!/bin/bash
x=66
test $x -eq 66
echo $?
```

```
#!/bin/bash
x=66
test $x -eq 10
echo $?
```

```
#!/bin/bash
x=66
[ $x -eq 10 ]
echo $?
```

```
#!/bin/bash
if [ "tizio" = "tizio" ]
then
    echo "Espressione valutata come vera"
else
    echo "Espressione valutata come falsa"
fi
```

```
#!/bin/bash
#
# Un piccolo gioco per tentare la fortuna
#
# Carlo A. Mazzone
```

```
clear
echo "-INDOVINA IL NUMERO-"
echo "Tenta la fortuna, gioca un numero: "
read giocata
numero=$RANDOM
echo "Il numero magico e': $numero "
if [ $giocata -eq $numero ]
then
    echo "INCREDIBILE! Hai vinto."
elif [ $giocata -lt $numero ]
then
    echo "Spiacente, il tuo numero e' inferiore."
else
    echo "Spiacente, il tuo numero è superiore."
fi
```

```
#!/bin/bash
if [ -a carlo.txt ] ; then
    echo "File presente"
else
    echo "Non trovo il file!"
fi
```

```
#!/bin/bash
PERCORSO="/home/carlo"
if [ -a $PERCORSO/carlo.txt ] ; then
    echo "File presente"
else
    echo "Non trovo il file!"
fi
```

```
#!/bin/bash
pluto=""
if [ -n $pluto ]
then
    echo "Lunghezza della stringa maggiore di zero."
else
    echo "La stringa è vuota!"
fi
```

```
#!/bin/bash
stringa1="Carlo"
stringa2="Carlo"
if [ $stringa1 = $stringa2 ]
```

```

    then
        echo "Le due stringhe sono uguali."
    else
        echo "Stringhe differenti!"
    fi

#!/bin/bash
stringa1="Carlo"
stringa2="Ludovica"
if [ "$stringa1" \< "$stringa2" ]
    then
        echo "La prima stringa precede la seconda."
    else
        echo "La seconda stringa precede la prima."
    fi

#!/bin/bash
arg1=10
arg2=12
if [ $arg1 -gt $arg2 ]
    then
        echo "$arg1 e' maggiore di $arg2"
    elif [ $arg1 -lt $arg2 ]
        then
            echo "$arg1 e' minore di $arg2"
        else
            echo "I due numeri sono uguali."
        fi

#!/bin/bash
#
# Ritenta la fortuna
#
# Carlo A. Mazzone

clear
echo "-INDOVINA IL NUMERO-"

risposta=s
while [ "$risposta" = s ]
do
    echo "Tenta la fortuna, gioca un numero (da 1 a 10): "
    read giocata
    numero=$RANDOM
    let numero=$numero%10+1
    echo "Il numero magico e': $numero "
    if [ $giocata -eq $numero ]
        then
            echo "INCREDIBILE! Hai vinto."
        elif [ $giocata -lt $numero ]
            then
                echo "Spiacente, il tuo numero e' inferiore."
            else
                echo "Spiacente, il tuo numero è superiore."
            fi
        #Chiedo conferma per continuare a giocare
        echo "Vuoi giocare ancora? Digita s per continuare."
        read risposta
    done

#!/bin/bash
for i in $( ls )
do
    echo "File: $i"

```

```
done

#!/bin/bash
for (( CONTATORE=1; CONTATORE<10; CONTATORE++ ))
do
    echo "Questa e' la riga $CONTATORE"
done
```

Capitolo 11 – Sviluppo software con Linux

Quando qualcuno dice, "Voglio un linguaggio di programmazione nel quale ho bisogno soltanto di dire ciò che voglio venga fatto", dagli un lecca-lecca.

Alan Perlis

```
#include <stdio.h>
int main()
{
    printf("Salve, mondo");
    return 0;
}

#include <iostream>
using namespace std;
int main()
{
    cout << "Salve, mondo.";
}

program SalveMondo;
begin
    writeln('Salve mondo');
    readln;
end.

class SalveMondoApp {
    public static void main(String[] args) {
        System.out.println("Salve mondo!");
    }
}

print ("Salve mondo da Python")

#!/usr/bin/env perl
print "Salve mondo dal Perl";
```

Capitolo 12 – Linux e la gestione dei processi

Non è la più forte delle specie che sopravvive, né la più intelligente, ma quella più reattiva ai cambiamenti.

Charles Darwin

```
#include <stdio.h>
#include <unistd.h>
```

```
int main()
{
    int pid;
    printf("Ciao, io sono la fucina dei processi\n");
    printf("Qui siamo prima della fork\n");
    pid=fork();
    printf("Qui siamo DOPO la fork\n");
    return 0;
}
```

```
#include <stdio.h>
#include <unistd.h>
```

```
int main()
{
    int pid;
    printf("Qui siamo prima della fork\n");
    pid = fork();
    if(pid== -1)
    {
        /* Errore */
        printf("Errore.");
    }
    else if(pid==0)
    {
        /* Eseguito dal processo figlio */
        printf("Io sono il figlio. Il mio PID e': %d\n", getpid());
    }
    else
    {
        /* Eseguito dal padre */
        printf("Io sono il padre. Il PID di mio figlio e': %d\n", pid);
    }

    printf("Qui siamo DOPO la fork\n");
    return 0;
}
```

```
#include <stdio.h>
#include <unistd.h>
```

```
int main()
{
    int pid;
    int x=5;
    printf("Qui siamo prima della fork\n");
    pid = fork();
    if(pid== -1)
    {
        /* Errore */
    }
```

```

        printf("Errore.");
    }
    else if(pid==0)
    {
        /* Eseguito dal processo figlio */
        x=6;
        printf("Io sono figlio. Il valore di x e': %d\n", x);
    }
    else
    {
        /* Eseguito dal padre */
        x=7;
        printf("Io sono il padre. Il valore di x e': %d\n", x);
    }

    printf("Qui siamo DOPO la fork. Il valore di x e': %d\n", x);
    return 0;
}

#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h> //necessario per la funzione wait

int main()
{
    int pid;
    printf("Esegui e aspetta\n");
    pid = fork();
    if(pid== -1)
    {
        /* Errore */
        printf("Errore.");
    }
    else if(pid==0)
    {
        /* Eseguito dal processo figlio */
        execl("/bin/ls", "ls", "-la", "/var", NULL);
    }
    else
    {
        /* Eseguito dal padre */
        wait(NULL);
        printf("Il figlio ha terminato.\n");
    }

    return 0;
}

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h> //necessario per la funzione exit

int main()
{
    int pid;

    printf("Qui siamo prima della fork. Il mio pid e': %d\n", getpid());

    pid = fork();
    if(pid== -1)
    {
        /* Errore */
        printf("Errore.");
    }
    else if(pid==0)
    {
        /* Eseguito dal processo figlio */
        printf("Io sono il figlio. Il mio PID e': %d e mio padre e': %d\n", getpid(), getppid());
    }
}

```

```

        sleep(5);
        printf("Io sono il figlio. Il mio PID e': %d e mio padre e': %d\n", getpid(), getppid());
    }
    else
    {
        /* Eseguito dal padre */
        printf("Io sono il padre. ");
        printf("Ora mi riposo per un secondo e termino.\n");
        sleep(1);
        exit(0);
    }

    printf("Qui siamo DOPO la fork. \n");
    return 0;
}

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

void *stampa(void *);

int main()
{
    pthread_t test_thread;
    int rc;

    rc = pthread_create (&test_thread, NULL, stampa, NULL);

    if (rc)
    {
        printf("Errore di pthread_create con codice %d\n", rc);
        exit(-1);
    }

    sleep(1); //diamo tempo per la visualizzazione del messaggio dal thread
    return 0;
}

void *stampa(void *argomento)
{
    printf("Stampa eseguita\n");
}

```

Appendice

Non smetteremo di esplorare. E alla fine di tutto il nostro andare ritorneremo al punto di partenza per conoscerlo per la prima volta.

T. S. Eliot

```

/*
Un esempio pratico
Autore: Carlo A. Mazzone
*/
#include <stdio.h>
int main()
{
    printf("salve, mondo");
}

```

```

}

/* variabili */
#include <stdio.h>
int main()
{
    int x;
    printf("Dammi un numero: ");
    scanf("%d", &x);
    printf("Il numero digitato e': %d ", x);
}

#define MAX_NUMERI_GESTIBILI 99
#include <stdio.h>
int main()
{
    /* istruzioni */
}

/* calcola il precedente e successivo di un numero inserito da tastiera */
#include <stdio.h>
int main()
{
    int x, prec, succ;
    printf("Dammi un numero: ");
    scanf("%d", &x);
    printf("Il numero digitato e': %d\n", x);
    prec=x-1;
    printf("Il numero precedente e': %d\n", prec);
    succ=x+1;
    printf("Il numero successivo e': %d\n", succ);
}

/* determina il maggiore tra due numeri */
#include <stdio.h>
int main()
{
    int a, b;
    printf("Inserisci il primo numero: ");
    scanf("%d", &a);
    printf("Inserisci il secondo numero: ");
    scanf("%d", &b);
    if (a > b)
    {
        printf("Il primo numero e' maggiore del secondo");
    }
    else
    {
        printf("Il secondo numero e' maggiore del primo");
    }
}

#include <stdio.h>
int main()
{
    int scelta;

    printf("Fai la tua scelta \n");
    printf("1: - Inserisci nominativo\n");
    printf("2: - Modifica nominativo\n");
    printf("3: - Cerca nominativo\n");
    printf("4: - Cancella nominativo\n");
    scanf("%d", &scelta);
    switch (scelta)

```

```

{
    case 1:
        printf ("Inserimento");
        break;
    case 2:
        printf ("Modifica");
        break;
    case 3:
        printf ("Ricerca");
        break;
    case 4:
        printf ("Cancellazione");
        break;

    default:
        printf ("Voce selezionata non valida.\n");
}
printf("\n");
return 0;
}

for (i=1; i<=10; i++)
{
    printf("Io sono la riga numero %d\n", i);
}

i=1;
while (i<=10)
{
    printf("Io sono la riga numero %d\n", i);
    i++;
}

int count=0;
int x;
printf("Digita i caratteri della sequenza - 0 per terminare\n");
do
{
    scanf("%d", &x);
    count++;
}while (x!=0); /* continuo il ciclo finché non trovo il carattere zero */

/* decremento di 1 il valore di count per escludere lo zero dal conteggio */
printf("Lunghezza sequenza: %d", count-1);

printf("\n");

int cambia(int);
int main()
{
    int a=1, b=0;
    printf("Il valore di a prima della chiamata e': %d\n", a );
    printf("Il valore di b prima della chiamata e': %d\n", b );
    b=cambia(a);
    printf("Il valore di a dopo la chiamata e': %d\n", a );
    printf("Il valore di b dopo la chiamata e': %d\n", b );
    return 0;
}

int cambia (int a)
{
    a=a+1;
    return a;
}

```

```

int cambia(int *);

int main()
{
    int a, b, *punt;
    punt=&a;
    a=1;
    printf("Il valore di a prima della chiamata e': %d\n", a );
    b=cambia(punt);
    printf("Il valore di a dopo la chiamata e': %d\n", a );
    return 0;
}

int cambia (int *punt)
{
    *punt=*punt + 1;
    return *punt;
}

```