# Nozioni di AS400 (IBMi)

L'AS400 è un mainframe (che funziona da server web, etc.) collegato a vari terminali emulati.
I caratteri sull'AS400 sono lunghi massimo 10 caratteri.
I campi di input-output sono sottolineati e si possono navigare usando la tabulatura. Un campo sottolineato bianco è obbligatorio, mentre gli altri solitamente hanno valori di default.

Nella prima schermata c'è `Sistema`, `Sottosistema` (entrambi indicati con Q sia di AS/400 sia dichiarati dall'utente. I sistemi e sottosistemi sono a lotti "batch").

I comandi sono oggetti di 10 caratteri: *verbo + (attributo) + predicato* tendenzialmente senza vocali:

- SHIFT + Fn - formula i numeri superiori al 12 fino al 24.

L'asterisco in AS/400 indica un valore simbolico e può servire per ricerca di oggetti o parametri di comando:

- ad esempio, se cerchi un file usando MYFILE*, il sistema restituirà tutti i file che iniziano con MYFILE, come MYFILE1, MYFILEA, ecc.
- `*ALL` : Utilizzato per indicare che si desidera considerare tutti gli oggetti o tutti i valori possibili in un determinato contesto (eg. `WRKOBJ OBJ(MYLIB/*ALL)`).

Molti comandi utilizzano acronimi che seguono una struttura ben definita per eseguire varie operazioni sul sistema

- `DSP` (Display. eg. `DSPJOB`, visualizza solo, non modifica)
- `WRK` (Work with. eg. `WRKLIB`, `WRKOBJ`) - visualizzare e interagire con gli oggetti
- `CHG` (Change. eg. `CHGJOB`, `CHGPWD` cambiare password)
- `DLT` (Delete)
- `CRT` (Create. eg `CRTJOBQ`, `CRTLIB`)
- `END` (End: termina)
- `SND` (Send)
- `STR` (Start)
- `RST` (Restore)
- `SAV` (Save)
  l'asterisco viene usato per specificare un valore speciale o una scelta predefinita: `*ALL`, `*NONE`, `*CURRENT` (contesto corrente o l'utente corrente), `*SYSVAL`.

# I tasti funzione

- **F1 - Help**: Mostra la guida in linea per la schermata corrente o il comando che stai utilizzando. Se il cursore si trova su una sezione dello schermo, mostra una una descrizione della porzione di testo.
- **F2 - Change**: Permette di modificare le informazioni nella schermata corrente.
- **F3 - Exit**: Esce dalla schermata corrente o annulla l'operazione in corso.
- **F4 - Prompt**: Mostra i parametri di input per il comando corrente, consentendo di inserire i valori richiesti.
- **F5 - Refresh**: Ricarica la schermata corrente, utile per aggiornare le informazioni visualizzate.
- **F6 - Create**: Permette di creare nuovi oggetti, come file, librerie, ecc.
- **F7 - Scroll Up**: Scorre verso l'alto nella schermata corrente.
- **F8 - Scroll Down**: Scorre verso il basso nella schermata corrente.
- **F9**: ripete l'ultimo comando.
- **F10 - Display/Hide Menu**: Mostra o nasconde il menu principale nella schermata corrente.
- **F12 - Cancel**: Annulla l'operazione corrente o torna alla schermata precedente.
- 3: copia

# GO

Il comando `GO` è utilizzato per accedere a un menu o una sottosistema di comandi di opzioni e comandi specifici che puoi eseguire.

- `GO` - selezione menù
- `GO MENU` - menù personale
- `GO PROGRAM` - menù per la gestione dei programmi
- `GO MAIN` - menu principale standard del sistema

# Gli oggetti: WRKOBJ

Un oggetto su AS/400 è qualsiasi elemento del sistema che può essere creato, gestito o utilizzato. Gli OBJ sono per esempio file, programmi, librerie e altre risorse, memorizzati permanentemente su disco:

- Libreria ( `*LIB` . Nota: le librerie non possono contenere altre librerie)
- File ( `*FILE` )
- Programma ( `*PGM` )
- Area dati ( `*DTAARA` )

- Coda di output ( `*OUTQ` )

Ogni oggetto è identificato da 3 oggetti di max 10 caratteri:

- nome
- topo
- posizione

`WRKOBJ` + F4 - menu per inserire parametri del comando senza bisogno di conoscerli a memoria (nome, libreria e tipo).

```
WRKOBJ OBJ(libreria/oggetto) OBJTYPE(*tipo)
```

- `OBJ` - nome dell'oggetto che vuoi cercare. * per cercare con un certo pattern.
- `OBJTYPE` - il tipo di oggetto (ad esempio, `*PGM` per programmi, `*FILE` per file, `*LIB` per librerie, etc.).

WRKOBJ GIULIOS mosta gli obj del profilo.
*QUSRSYS tipo libreria
LIBL sta per library list

# I Jobs

I **job** (o task) su IBMi (AS/400) sono processi (unità minima di elaborazione) o attività che il sistema esegue.
Ci sono diversi tipi:

- **Job interattivi** (iniziano con `INTER` . Richiedono l'interazione dell'utente. Per esempio il login di un utente)
- **Job batch** (per operazioni programmate o lunghe. Lanciati dalla Job Queue. Se eseguo un programma video nel sottosistema batch, darà errore)
  - `Qbatch` ( `batch` =lotto. Ripete in loop un certo numero di operazioni costanti in cui non è necessario un operatore).
- **Job di sistema** (iniziano con `Q` )
  - `QINTER` è un sottosistema che gestisce job interattivi con altri utenti, quindi 'compila' la richiesta
- **Job di comunicazione** (trasferimento dati (come FTP o SNA))
- **Job di spool** (stampanti o altri dispositivi di output)

`WRKACTJOB` mostra i sottosistemi

- Stato `MSGW` vuol dire che è in attesa di messaggio

Componenti di un Job:

- Job Name
- Job Number (Numero univoco assegnato automaticamente dal sistema. Progressivo che poi riparte dall'inizio)
- User (utente che ha avviato il job o sotto il quale è eseguito)
- Job Queue (in che ordine e quando sarà eseguito (soprattutto per i job batch)).
- Job Description (percorsi del file, le librerie da usare e altre risorse).

`signoff` - serve per deloggarsi da terminale e termina i lavori

Diversi stati del job:

- `ACTIVE`
- `WAITING`
- `HELD` (pausa)
- `COMPLETED`
- `ENDED` (terminato manualmente o automaticamente)

`DSPJOB` : È principalmente per visualizzare informazioni dettagliate su un job. È un comando di sola lettura che fornisce una panoramica dettagliata del job.
`WRKJOB` (visualizzazione del lavoro), al contrario di DSPJOB (per visualizzare informazioni dettagliate su un job), è utilizzato per gestire i job: visualizzare un elenco di job, eseguire operazioni, e gestirli.

# lista delle librerie

Una libreria è un oggetto che contiene altri oggetti, ma non altre librerie.

La lista delle librerie il path che il sistema utilizza per ricercare gli oggetti.
Il comando `DSPLIB` mostra le librerie.
La lista delle librerie è locale ad ogni utente ed è organizzata su 3 livelli:

- lista delle librerie di sistema ( `QSYS` , `QCMD` , etc.)
- lista corrente ( `CUR` , corrente, è la prima delle librerie. Potrebbe tranquillamente non esistere. Viene usato per buttare altre librerie di scarto)
- lista delle librerie utente

`EDTLIBL` - per modificare l'elenco delle librerie. I numeri sono di 10 in 10 per spostarle nell'elenco.
Per togliere una libreria dall'elenco basta cancellarne il testo.

Sullo 0 si aggiungono librerie.

- `QTEMP` : temporanea
- `QGPL` : (general purpose library) usato per copiare alcune informazioni che potrebbero venir eliminate

Il formato dei caratteri usato è la BCD (Binary-Coded Decimal): ogni cifra decimale viene convertita singolarmente nel suo equivalente binario a 4 bit.

`ADDLIBLE (libl entry)` - per aggiungere una libreria all'elenco delle librerie

```
ADDLIBLE LIB(TESTLIB) POS(*AFTER)
```

`RMVLIBLE` - rimuove la libreria

CTRL+INVIO = uscita campo

`DSPJPBLOG` + F10 + shift F6: vedere il log.

# librerie

Le liste delle librerie non si aggiornano automaticamente
`CRTLIB (Create Library)` crea una nuova libreria.
`DLTLIB` rimuove una libreria

# Gli oggetti

Fra i file fisici di tipo SRC (PF-SRC) compariranno sempre questi 4 file:

- `QPFSRC` , contiene i sorgenti dei file fisici, ovvero la struttura dei file.
- `QLBLSRC` , contiene i sorgenti che serviranno per creare i programmi Cobol.
- `QCLSRC` , contiene i sorgenti che serviranno per creare programmi di tipo CLP.
- `QDDSSRC` , contiene i sorgenti che serviranno per creare le maschere video (DSPF) e i PRTF (file di stampa).

gli oggetti:

- `PGM` sono COBOL CMP
- `*FILE`
  - `DF` - DISPLAY FILE (Maschere video)

        - in `QDDSSRC`
    - `PF` - PRINTER FILE
        - in `QPFSRC`
    - `PF` - PHYSICAL FILE
    - `LF` - LOGICAL FILE

# PDM

I membri della libreria:

- Programma di sviluppo `GO PROGRAM` + 2 o `strpdm` (start+pdm)
- `CRTSRCPF FILE(utente/file)` Creazione file sorgenti.
- creiamo il file `QPFSRC` e `QCLSRC` (control language)
- F5 -> aggiorna

Il **PDM (Program Development Manager)** è uno strumento utilizzato principalmente per gestire e modificare file sorgente, oggetti di librerie e processi di compilazione.

Il **SEU (Source Entry Utility)** è un componente del PDM, e costituisce un editor di testo per il codice sorgente.

Se digito a sinistra il numero, vado alla riga interessata

- `I` + INVIO + testo -> nuova riga (posso mettere un numero per dire quante righe)
- `C` copy
    - `CC` copia un gruppo
    - `a` (after) per incollare dopo (non deve essere ad inizio)
    - `b` (before) per incollare prima
- `D` delete
- `M` move
    - `a` (after) per incollare dopo
    - `b` (before) per incollare prima

su SEU (F10) se metto:

- `SAVE` salva
- `F` +nome o nome+F16 cerca
    - si può usare anche SHIFT+F4
- `C` +nomeprec+nomenuovo change
    - `C` +nomeprec+nomenuovo `ALL` cambia tutti

    - `C` nprec nnuovo `a`
- `rp` repeat: ripete la riga

Shortcuts:

- `F5` ricarica
- SHIFT+F8 date di modifica.
- SHIFT+F2 apri secondo file

```
XX
  XX
```

Nasconde il testo e `SF99` e `SL99` fanno vedere il testo. `F5` fa rivedere tutto.

# DDS

I **DDS (Data Description Specifications)** sono un tipo di codice viene utilizzato per definire file fisici, file logici e altri oggetti di database sull'AS/400.

```
----.----1----.----2----.----3----A----4----.----
        *-------------------------------------------------*
        *---Procedura - Modulo base                  ---*
        *---Area      - Formazione                   ---*
        *---Archivio  - Archivio sequenziale         ---*
        *-------------------------------------------------*
COMME                                    UNIQUE
     A          R FILEREC
     A            FILEDAT1      10A     ALIAS(FILE_DATO_1)
     A                                    TEXT('DATO 1')
     A                                    COLHDG('DATO 1')
     A          K FILEDAT1
```

Un commento viene dichiarato con `*---` e non deve essere necessariamente chiuso.

- Procedura: l'argomento della release. Serve per distinguere se verrà utilizzata o meno da ogni cliente in base ai moduli acquistati (eg. tipo di finanziamento);
- Area: contesto;
- Archivio: Tipo di file

Il codice:

- I primi 5 spazi dopo il numero della riga è uno spazio per i commenti
- `UNIQUE` (colonna 45) indica che un archivio è a chiave unica
- `A` (colonna 6) indica che quella riga è attiva e verrà elaborata dal sistema.
- `R` (colonna 17): segnala che la riga definisce formato o nome (max 10) di record
- `K` archivio chiave ordinato dal campo chiave FILEDAT1

Ci sono 3 campi obligatori:

- nome
- la natura del campo:
  - `10A` dice 10 campi alfanumerici (Maiuscolo è diverso dal minuscolo)
  - `10S` 10 numerici. L'ultimo bite aggiuntivo è il segno
  - `5S 2` lungo 5 con 2 decimali

3 facoltativi:

- `ALIAS` : un identificativo più lungo di 10 caratteri. Non va usato lo spazio, ma l' _ .
- `COLHDG` : descrizione che viene visualizzato in interfacce visuali
- `TEXT` : descrizione, vale come un commento

`crtpf` + F4 -> creazione file fisico (CRTPF).
F10 apre impostazioni avanzate

- oggetto nome (sorgente/membro e oggetto devono avere nomi uguali)
- in che libreria ( `GIULIOS` )
- file origine è la cartella ( `QPFSRC` )
- in membro origine `*FILE` si riferisce al nome file
- numero massimo di membri: `*nomax`
- numero iniziale record: `*nomax`
- tempo massimo di attesa record: `*nomax`
- riutilizzo record cancellati: `*YES`
- controllo livello form record: `*no`

**OCSSY401** è una libreria dove ci stanno file cobol e sorgenti

Per consultare il contenuto di un archivio:

- `upddta`
  - f10 immissione
  - col tasto PAG si scorre un file
  - SHIFT+F11 -> elimina un campo

- `wrkf` per visualizzare un file.

In **bianco** è indicato un dato chiave che decide l'ordine (se è letterale è ordinato per numero)

Anche gli archivi fisici con una chiave hanno un sorgente nella `SPFSRC`

# spool

Spool è la stampa di ogni procedura eseguita (Compilazione, etc.).
`wrksplf` / `WRKSPLF SELECT(GIULIOS)` / `SP` sul file porta allo spool.

Gli errori dal 30 in su impediscono la compilazione del programma.

# Dizionario

Il dizionario è un oggetto privo di membri ed è sempre nel `QPFSRC` .
Esempio:

```
A          R DIZIONARIO
A            CAMPO_TIPO1    10A    COLHDG('Tipo 1')
A            CAMPO_TIPO2     5S 0  COLHDG('Tipo 2')
              *--- 5 CIFRE 0 decimali
A            CAMPO_TIPO3    30A    COLHDG('Tipo 3')
```

Per riferirsi a un dizionario:

```
A                                UNIQUE
A          R PRINCIPALE          REF(DIZIONARIO)
A            CAMPO1              REFFLD(CAMPO_TIPO1)
A          K CAMPO1
```

- `REFFLD` (Ref-field) indica un campo nel dizionario.
- `REF(DIZIONARIO)` - sotto unique

Per compilare il dizionario: `crtpf` membro su `*none`

Membro se desiderato su `*FILE` in tutti gli altri casi

`DSPOBJD` (Display Object Descriptions, Tasto 8) : Per sapere un oggetto a quale sorgente punta bisogna guardare `File origine` compilando:

- oggetto: `FILE1`
- tipo oggetto: `*FILE`

# Denominazioni

- NOME File fisico: `FOcnFxxx`
- RECORD: `FOcnxxxREC`
- CAMPO/colonna: `FOcnxxxaaa`
  - aaa è il nome del campo
- ALIAS: `FOcnxxx_CODICE`
- NOME File logico: `FOcnFxxxXn`

ocssy401

INCV

- modulo intermediari
- divisione successiva
- `F` è un file fisico. `V` è Maschera video, etc.
- NUM = numeratori (cosa contiene)

# importare record da altri file

Per importare un record da un altro file, posso usare il comando `FORMAT` :

```
                  UNIQUE
R NUOVO_RECORD    FORMAT(NOME_FILE_ORIGINE)
K CAMPO_1
K CAMPO_2
```

NUOVO_RECORD stesso nome di record nel file di origine

# Viste logiche (LF)

Permette di avere viste diverse di un file. Crea una sorta di puntatori a righe specifiche di codice che soddisfano determinate condizioni.

```
A                                       UNIQUE
A           R NUOVO_RECORD              PFILE(NOME_FILE_ORIGINE)
A           K CAMPO_1
A           K CAMPO_2
A           S CAMPO_1                   COMP(NE 0)
```

Se metto `UNIQUE` nella vista logica, si riflette nel file.
FORMAT
`COMP` (compare) dice una condizione che dice NOT EQUAL 0. Invece che `S` si può usare `O` (omit)

- `GT` : >
- `LS` : <
- `LE` : <=
- `GE` : >=

`crtlf` (o numero `14` )

- numero massimo di membri: `*nomax`
- tempo massimo di attesa record: `*nomax`
- controllo livello form record: `*no`

Devo cancellare le viste logiche per cancellare il file
`DSPDBR` (Display database relations)

`CV` confronta membri

# SQL

`strsql` (Start SQL) + F4

F15 -> va a capo

Gli importi vanno in centesimo di euro.

- `SELECT RTRIM(Nome) CONCAT ' ' CONCAT Cognome FROM Tabella WHERE Value IN (1, 3, 5)`
- `BETWEEN 1 AND 2`
- `SELECT UPPER(nome) AS nome_maiuscolo FROM clienti;`
- `LIKE '%a'`

# Funzioni

- `AVG(campo)`
- `SUM()`
- `LENGTH()`
- `SUBSTR(Valore, 3, 2)` (parte da 1)
- `LOWER()` / `UPPER()`
- `ABS()`
- `DIGITS` / `DEC()`
  - Persona CONCAT Digits(Numero)
- `REPLACE(Nome, 'IN', '01')`
- `SIGN()`
- `CAST(Nome AS CHAT(20))`
- `CONCAT(CONCAT(Nome, ' '), Cognome)`
- `WHERE Campio IS NULL` / `HAVING`

```
CREATE FUNCTION
  Iva(@prezzo DECIMAL(10, 2), @tasso DECIMAL(5,2))
  RETURNS DECIMAL(10,2)
AS
BEGIN
  RETURN @prezzo * tasso / 100
END;


SELECT dbo.Iva(100, 2)
```

# Subquery

```
  WHERE EXISTS
```

# Inner JOIN

```
  SELECT * FROM GUALTIERO/LIBRI1 JOIN GUALTIERO/LIBRI2 ON LIBRI1.FOAM_LIBRO = LiBRI2.FOAM_LIBRO
```

- LEFT JOIN: fa vedere tutti quelli nel file di sinistra

## Creare tabelle

```
CREATE TABLE Lib/Nome
(CAMPO1 CHAR(19),
CAMPO2 NUMERIC(3)
CAMPO3 NUMERIC(5))


DROP TABLE QTEMP/FILE1
```

Crea un campo FISSO

```
SELECT 'FISSO' AS FISSO FROM GUALTIERO/LIBRIQ


CREATE TABLE QTEMP/LIBRIW AS (SELECT * FROM GUALTIERO/LIBRI1) WITH NO DATA


CREATE TABLE QTEMP/LIBRIW LIKE GUALTIERO/LIBRI1


INSERT INTO QTEMP/LIBRIQ
VALUES(8, 'KING', 'STEPHEN', 'THE STAND', 'FISSO lUNGO')


INSERT INTO QTEMP/LIBRIQ (SELECT [...])


UPDATE INTO QTEMP/LIBRIQ  SET FISSO = 'FISSO3' WHERE NOME = 'PLUTO'


DELETE FROM QTEMP/LIBRIQ WHERE FISSO = 'FISS4'
```

Come creo senza usare LIKE?

# COBOL

Il COBOL è in `QLBLSRC`

```
01 <alias nome-file>-REC
```

Con `COPY DDS-ALL-FORMATS` verrà esattamente copiata la struttura del FILE1:

```
DATA DIVISION.
FILE SECTION.
FD FILE1.
01 FILE1-REC.   COPY DDS-ALL-FORMATS OF FILE1.
```

L'utilizzo di DD-ALL-FORMATS implica che i nomi utilizzati nel programma saranno gli ALIAS del file, mentre l'utilizzo di DDS-ALL-FORMATS è preferito e implica che i nomi utilizzati nel programma saranno i nomi short.

```
B01-INPUT.
  OPEN INPUT FILE1.

  INITIALIZE FILEREC.
B01-READ.
  READ FILE1 NEXT
    AT END
      GO TO B01-CONTINUA
  END-READ.

  GO TO B01-READ.

B01-CONTINUA.
  CLOSE FILE1.

B01-EX.
  EXIT.
```

```
D01-IO.
  OPEN I-0 FILE1.

INITIALIZE FILEREC.
  D01-READ.
    READ FILE1 NEXT
  AT END
    GO TO D01-CONTINUA
  END-READ.

  MOVE 1      TO FILE-DATO-2.
  REWRITE FILE1-REC.

  GO TO D01-READ.

D01-CONTINUA.
  CLOSE FILE1.

D01-EX.
  EXIT.
```

COSTO

QCLBLSRC
QBLSRC le copie personali

B01-PRELIMINARI.
CALL 'OTTSPLDA' USING LDA-REC BY CONTENT ' '.
OPEN I-O VIDEO.

`RCLSRC` (Reclear Resource) Spesso aggancia i test all'oggetto precedente.

La copy cobol `OXDBYREF` è una copy cobol dizionario in quanto contiene le variabili più comuni e più utilizzate nel sistema OCS e quindi dichiarata in moltissimi programmi.
È l'equivalente di `OCSDBREF` per i file.

I test vanno fatti in `WORKOCS` o `DEMOOCS`

`CC` COMPILA
`CR` TESTA

# Maschere video

Le maschere sono in `QDDSSRC` di tipo dspf di tipo *FILE.

Per compilare: `CRTDSPF` crea le maschere, ma non si usa. Si usa `CC` .

Per modificare i file video:

- `CHGDSPF` + F4
- `2` : modifica
- `17 -> 12` : modifica a video

`WRKOCS`

ASSUME (di tipo RECORD) permette

- opzione 8
  - dimensione della finestra
  - riga corrente: parte da riga 3
  - posizione corrente: parte da colonna 7
  - righe finestra (MAX 20, DEFAULT 4)
  - posizione finestra: (MAX 74) larghezza di 64
- opzione 12 si edita

E' composta da:

- eichetta: `'Campo 1'`
- variabile: `+B(10)`
  - alfanumerici: ```
    - `B` editabile
    - `O` non editabile
  - numerico
    - `9` editabile
    - `6` non editabile

Non bisogna mischiare variabili di Input e di Input-Output, ma bisogna metterli su protetti.

f10

editing campi

- `-` sposta
- `==` per copiarlo

- `d` cancellare
- `ac`
- SHIFT+F8 per evidenziare i campi
- `?` da info sul campo
- `*`
    - `attributi di visualizzazione` :

Nel login:

`WORKOCS`

`WORKOCS0`

gestione controparti:

- campi
    - alta intensità il campo è bianco
    - l'inversione di fondo si attiva quando si scatena qualche condizione
- Y SI imposta per l'attributo
- l'indicatore ha valore numerico a due cifre ( `01` - `99` )
    - `+` invio
    - stessa riga: `AND`
    - opposta: `OR`
    - quando c'è un errore si disattiva il sottolineato

- `F6 + Y` per selezionare i campi da accendere
- F4 per riordinare i campi
    - F6 per riordinare i campi
- F20 (SHIFT+F8) accende le etichette

`CALL` è per eseguire un programma

```
SELECT VIDEO    ASSIGN
```

`CPINTDCM1` FM

in parole chiave comuni
posso dichiarare l'alias.

FMST_ERRORE

INIZIALIZE: preparo la schermata pulita. COsa fa se WRITE la pulisce?

```
IF W-FUNKEY = 7
  GO TO A01-FINE
END-IF.
```

GLi dice che se qualcuno preme F7, questo deve fare.

```
PERFORM Nome_funzione THRU Posizione_finale_funzione
```

CPINTTCA
CPINTTCAFM

sui campi variabili i valori dall'1 al 60
80-999 sui tasti funzionali.

'I' Immissione
'SPACE' inserimento

Routine per controllare un parametro:

```
CALL 'SYCNFC' using
    BY CONTENT '0024'
    BY REFERENCE  LKMSG-PAR
                  LKMSG-MSG
MOVE LKMSG-MSG               TO FMST-ERRORE
```

Tabella file status: `CALL SYFILSTA`

`rcmsg` -> per vedere i codici dei messaggi di errore (eg. `0548` : nome e cognome obbligatorio)

`8 e 5` dettagli sul compilato

```
  WORKING STORAGE SECTION
  COPY OXDBYREF. *--- copia il dizionario di OCS
    01 LDA-REC.          COPY CTLDA (local data area) *--- contiene informazione di sistema
  COPY CTANALOCRT.
  OTSSPLDA
```

in WORKOCS
DSPDTAARA *LDA mostra le info di LDA

`WK-` identificano variabili d'appoggio

COMP-3. la metà +1

```
SW-TESTATA      PIC(9)
  SI-TESTATA    VALUE 1
  NO-TESTATA    VALUE 0
```

`wrkobj` devo controllare l'ora
`CR` creo solo lo spool

`RC` per capire da cosa è chiamato ( `COPY` ) un programma

- Nel proprio profilo:
  - `DG` per debuggare
  - F6 do il punto iniziale
  - F12 do l'ultimo punto
  - con F10 vado avanti nell'esecuzione
  - F11 EVAL variabile = 'valore'
- In WORKOCS:
  - F1
    - prendo i miei campi che interessano
  - F10
  - `edtlibl` per aggiungere la propria libreria
  - CALL nome_programma
  - `enddbg` per terminare il debug
    - `ENDSRVJOB`
- SHIFT+ `F2` per agganciare il file che voglio debuggare nel mio file.
  - `F22` ( `SHIFT` + F10) per debuggare l'altro programma chiamato

# I messaggi di errore

Gestione dei messaggi di errore
`RCMSG` per cercare un errore

I messaggi di errore sono in `QPFSRC/OCCSSY401` `OTTSFMSG`

Il codice è un alfanumerico da 4 byte

0263 è per mettere i vzlori ammessi in un campo variabile nel programma.

I messaggi personalizzati sono in OTTSFMSG

```
MOVE 'T1/T2'              TO LKMSG-PAR
CALL 'SYMSGCL' USING
  BY CONTENT '1234'
  BY REFERENCE  LKMSG-PAR
                LKMSG-MSG
MOVE LKMSG-MSG            TO FMD-ERRORE
GO TO C01-VIDEO
```

Con la prima move dico che parametri stampare.
`RC -> MSG0155` : cerca 0155 di tipo *MSG

Per i messaggi di warning ci usa la routine `SYMSGWRN` .
Richiede

```
01 LK-CODICE      PIC X(4).
01 LK-PARAMETRI.
  03 LK-MODALITA        PIC X
  03 LK-CONFERMA        PIC X
  03 LK-STATUS          PIC XX
  03 LK-PARAMETRO       PIC X(20).
```

Chiamata:

```
CALL 'SYMSGCL' USING
  BY CONTENT '1234'
  BY REFERENCE  LK-PARAMETRI
```

`QPFSRC/OCCSSY401  SYNCHK`

`PIC X OCCURS 20.` -> crea un campo X e lo ripete. utile per il casting a numerico.

```
01 FMST-INTERM       PIC Z(9)
```

```
MOVE FMST-INTERM
CALL 'SYNCHK' USING
```

I FLAG SI/NO

OCSDBREF

Guarda il file `FRANCESCAG/QLBLSRC` , `CPPRVAOB`

Da file a video: `SYITNSNFV`

# Gli importi

DIchiarati a video come alfanumerici di 17.
A file è un campo numerico di 13.
`SYIMPFIVI`

CALL SYIMPFIVI USING WK-IMPORTO
FMS-IMP-RAGG
BY CONTENT LENGTH OF FMS-IMP-RAGG.

SYIMPNORM

# Le date

Vengono salvate in formato lungo invertito: anno-mese-giorno

SYDATK

CTLDA per la data odierna

SYDATFIST

SYDATFIAL
SYDATALFI
SYDATS -> calcolo data da numero a stringa
SYDATN ->

`DSPDTAARA *LDA` in WORKOCS recuperi tutte le informazioni.

# Modifica degli archivi

- `CPYF`
  - sostituzione aggiunta record
    - F1
  - F10
    - controllo campi formato record: `*NOCHK`

        - Se ho un numerico che va in un alfanumerico, funziona. alfa a num non va
    - `*MAP`
    - `*DROP` Quando alcuni campi non esistono in quello destinazione
- `CHPF` Fa una copia su se stesso e usa MAP e DROP.
  Copio un archivio in un'altro

# Programmi di controllo

`OXGAPTGAK` - Controlla l'esistenza del record in un file

```
INIZIALIZE OXGAPTGAK OF
```

# Gli elenchi

Subfile è l'elenco

ESC -> `16` -> `1` : per visualizzare o editare un campo

Colonna:

- Attributo (H) -> alta intensità
- Dsf Display Format -> specifico il formato
    - DAT -> data
    - IMP -> importo
      Key Return dice all'SQL che campo voglio restituire alla prossima schermata
      In tasti funzionali metto R per abilitare l'uso dell'invio. Se lo metto su L invece di F, funziona su quella specifica riga

OTSQPVLISTI
Libreria da dichiarare nella Working Storage Section: `OTSQYVLIST` .

```
WORKING-STORAGE SECTION.
OTSQYVLIST
```

F024FLIB

# I CL (Control Language)

Serve per automatizzare operazioni che si devono ripetere con una certa frequenza.

I programmi vengono testati in un ambiente separato (eg. `WORKOCS` )
I CL vengono raccolti in `QCLSRC`
Comando `STRSEU` :

- Tipo: `CLP`
  Lo script:

```
 ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+...
            PGM
            ADDLIBLE   LIB(GIULIOS)
            MONMSG     MSGID(CPF2103) EXEC(GOTO CMDLBL(FINE))
            CAL        PGM(FOAMLIB)
/*          RMVLIBLE   LIB(GIULIOS)                                */
 FINE:      ENDPGM


            MONMSG     MSGID(CPF2103) EXEC(DO)
                                        RMVLIBLE   LIB(GIULIOS)
                                        GOTO  CMDLBL(FINE)
                                       ENDDO
```

Con F4 si riordina il codice

`MONMSG` ignora l'errore `CPF2103`
Con FINE: si abbassa il livello di gravità per cui non ci sono variabili nel codice

Nel dumb (o eccezione)

- I ignore
- R retry
  SHIFT+ESC -> `3`

`EXEC` dice che se si verifica l'eccezione, salta tutto

`CPF0000` ignora tutti i messaggi di quella famiglia

## le variabili in un CL

DCL declare

```
...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+...
            PGM         PARM(&LIB &PGM)
            DCL         VAR(&LIB) TYPE (*CHAR) LEN(10)
            DCL         VAR(&PGM) TYPE (*CHAR) LEN(10)
            DCL         VAR(&MSG1) TYPE (*CHAR) LEN(40)
            DCL         VAR(&MSG2) TYPE (*CHAR) LEN(40)
            DCL         VAR(&MSG3) TYPE (*CHAR) LEN(40)
            DCL         VAR(&ERRORE) TYPE (*CHAR) LEN(30)
            CHGVAR      VAR(&ERRORE) VALUE('Dati di input errati!')
            IF          COND((&LIB *EQ ' ') *OR (&PGM = ' ')) THEN(DO)
                            CHGVAR  VAR(&MSG2) VALUE(&ERRORE)
                            CALL PGM(STOPMSG) PARM(&MSG1 &MSG2 &MSG3)
                        ENDDO
            ELSE        CMD(DO)
                        ADDLIBLE    LIB(&LIB)
                        MONMSG      MSGID(CPF0000)
                        CALL        PGM(&PGM)
            ENDDO
FINE:       RCLSRC
            ENDPGM
```

CALL GIULIOS/PROVAVAR ('GIULIOS' 'FOAMLIB')

# Le stampe

WORKOCS
Se sono 80 colonne con massimo 198 sta in A4
1 -> *PDF

SELECT STAMPA ASSIGN PRINTER-STAMPA80

## La testata

Dopo 60 righe c'è il salto pagina

WRITE ST-REC BEFORE ADVANCING.
ST-REC è una riga vuota

## Le stampe esterne

L'OVERLAY è la filigrana

```
WRITE STAMPA-REC FROM TESTATA-REC FORMAT 'DESC'
  AT EOP
    MOVE 'SI' TO                  TO WK-TESTATA
```

## Creazione di un printer file

`CTRSEU` in `QDDSSRC` di tipo `PRTF`
`19` (RLU - Report Layout Utility) -> F4 -> `larghezza:  80`

- `DR` : define record
- F23
  - SKIPA/SKIPB
  - SPACEA/SPACEB
- `DC` (Declare constant): creo un campo
  - `CF` (Center Field): centra il campo

```
<...>
-----
```

F13 F15
F11 definizione campo

SAVE mantiene una copia sul si

GIOVANNIg

INSERT?

SHIFT-F11 proprietà

SHIFT-F1 selezione multipla

Perché INITIALIZE-REC lascia i campi sporchi invece che scrivere INITIALIZEREC?

`wrkjob -> 4` per vedere le stampe

# versionamento e rilascio: le Release

OCSDB401 serve per compilare i programmi direttamente dal cliente
In WORKOCS numero 15

La release va aperta quando si modifica una tabella, non un cobol.

ZZnumero progressivo01

Le release vanno aperte col 5

```
       PGM
       SYUPDF FILE(nome_file) TIPO(*MAP)
       DLFT FILE(nome_file)
       MONMSG MSGID(CPF0000)
       SYCRTLF FILE(nome_file)
FINE:  RCLSRC
       ENDPGM
```

DLFT FILE(File_1) cancella vista logica
Il sistema aggiorna l'archivio e se non esiste lo crea

se aggiorniamo, meglio cancellare e ricreare

Non deve essere in linea l'OCSDB401.

Il tipo *NOCHECK non controlla e sovrappone

```
INITIALIZE FSTTIBR
[...]
WRITE FSTTIBE-REC
  INVALID KEY
    CONTINUE
END-WRITE.
```

Per sapere le viste logiche collegate: `DSPDBR`

SELECT * FROM SYRELEASE

QLBLSRC/OCSSY401 dove ci sono le subroutines che si possono chiamare

`RC` per cercare esempi di uso

- `Q-INTER` :
  - qualunque call va in QINTER
- `Q-BATCH` (lotto): atività fatta a gruppi.
  - Un programma in batch non deve avere un video

- `SBMJOB` (Immissione lavoro - Submit Job)
  - Con `WRKACTJOB` verifico se è in esecuzione
- gestione lavori inoltrati
- `WRKSBMJOB` per vedere se è stato eseguito

Uno script che ritarda il lavoro di un numero di secondi `SYDLY60`

```
*-----------------------------------------------------------------
WORKING-STORAGE SECTION.
01  WK-NUMREC                  PIC S9(10).
     EXEC SQL
        INCLUDE SQLCA
     END-EXEC.
*-----------------------------------------------------------------*
LINKAGE SECTION.
01  LK-PARAMETRI.
     03  LK-EMETTITORE             PIC S9(03).
     03  LK-TIPO-INTERMED          PIC X(03).
     03  LK-CATEGORIA              PIC X(03).
     03  LK-STATUS                 PIC X(02).
*-----------------------------------------------------------------*
PROCEDURE DIVISION          USING LK-PARAMETRI.
*-----------------------------------------------------------------
*-----------------------------------------------------------------*
PROCEDURE DIVISION          USING LK-PARAMETRI.
*-----------------------------------------------------------------
A01-PRELIMINARI.
     INITIALIZE                      WK-NUMREC.
     MOVE 'NT'                    TO LK-STATUS.
     EXEC SQL
       SELECT COUNT (*) INTO :WK-NUMREC
       FROM CRPROVISTB
       WHERE CRPTB_EMETTITORE    = : LK-EMETTITORE    AND
             CRPTB_INTERMEDIARIO = : LK-TIPO-INTERMED AND
             CRPTB_CATEGORIA     = : LK-CATEGORIA
     END-EXEC.
```

```
*-----------------------------------------------------------------*
B01-ELABORAZIONE.
     EXEC SQL
       DECLARE CARTE CURSOR FOR
         SELECT *
           FROM CRCAR
           WHERE CRCAR_KEY_N
                 BETWEEN :WK-CARTA-DA-N AND :WK-CARTA-A-N AND
                 CRCAR_PAG_FORMA = "TS" AND
                 CRCAR_ANA_AMMIN
                 BETWEEN :WK-AZIENDA-DA AND :WK-AZIENDA-A
     END-EXEC.

*--- Apertura cursore.
     EXEC SQL
       OPEN CARTE
     END-EXEC.
     CALL 'OTSQPLG'              USING SQLCA.

B01-READ.
    EXEC SQL
      FETCH NEXT FROM CARTE INTO :CRCAR-SQLREC
    END-EXEC.
    CALL 'OTSQPLG'              USING SQLCA.
    IF SQLCODE = 100 OR
       SQLCODE < 0
       GO TO B01-CLOSE
    END-IF.
```

# Il nuovo standard

```
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
      *--- Archivio tabella utenti collegamento remoti.
db-acc COPY BAUSERCOR IN FSELECT
db-acc      REPLACING    == :FILEALIAS: == BY == BAUSERCOR ==
db-acc                   == :FILEDB: == BY == BAUSERCOR ==.
       DATA DIVISION.
       FILE SECTION.
db-acc COPY BAUSERCOR IN FDD. DDS è per i file corti
      *--- uso il nome dell'ALIAS

IF FLAG
[...]

db-acc COPY FPROCEDURE
```

- COPY manda un comando al precompilatore per creare il sorgente
- BAUSERCOR il nome del sorgente del file che vogliamo utilizzare
- FSELECT dice che è una copy di selezione indicizzato
- FSELSEQ sequenziale
- REPLACING è per sostituire tutti i caratteri per creare nuove variabili
- FILEDB è il file sul disco che deve andare a leggere

```
IF FLAG-OPEN NOT = 'S'
   PERFORM OPEN-INPUT-BAUSERCOR
      THRU END-OPEN-INPUT-BAUSERCOR
   INITIALIZE   SYJOBUSER-PARAMETRI
CALL 'SYJOBUSER'  USING SYJOBUSER-JOB
                                          SYJOBUSER-USER
                                          SYJOBUSER-NBRJOB
  MOVE 'S'   TO FLAG-OPEN
END-IF.

MOVE 'OK' TO LK-ESITO.
MOVE SYJOBUSER-USER TO WK-UTENTE.
```

Chiusura:

```
PERFORM CLOSE-BNAIWINPUT
TARU END-CLOSE-BNAIWINPUT.


INITIALIZE BNCBTIFREC OF BNCBFTIF-REC.
MOVE LK-ABI TO BNCBTIF-ABI.
MOVE LK-CAB TO BNCBTIF-CAB.
MOVE LK-CONTO-CORRENTE TO BNCBTIF-CONTO-CORRENTE.
PERFORM READ-BNCBFTIF
    THRU END-READ-BNCBFTIF
  IF R-BNCBFTIF-INVALID-KEY
      IINITIALIZE BNCBTIFREC OF BNCBFTIF-REC
      MOVE 'NT' TO LK-STATUS
END-IF.
```

# I cicli descending

`START WITH NO LOCK` / `READ NEXT WITH NO LOCK` - perché di solito blocca quel record.

Se ho la chiave descending devo invertire START con END