Consiglio Nazionale delle Ricerche

Batch OS

R. Medves

93

CNUCE

Divisione Servizio Elaborazione Dati

A cura di : Riccardo Medves

Ristampa Luglio 1985
by - CNUCE - Pisa
Istituto del Consiglio Nazionale delle Ricerche

Indice dei capitoli

Introduzione

Questo manuale comprende un insieme di informazioni che interessano gli Utenti del sistema operativo OS presso il CNUCE: tali note presuppongono gia', da parte dell'Utente, una certa conoscenza generale del sistema operativo OS e del relativo Job Control Language.

Nella codifica del testo abbiamo usato le seguenti convenzioni:

- Le parole chiave (da codificare necessariamente nella forma scritta) sono mostrate a lettere maiuscole
- Le lettere minuscole indicano che deve essere inserito al loro posto un parametro
- Numeri e segni di punteggiatura vanno codificati cosi' come scritti
- Il segno ␣ indica lo spazio bianco

Il sistema operativo CS/SVS-HASP e' stato sostituito dal sistema OS/MVS-JES2: le funzioni svolte dalle componenti MVS e JES2 sono pero' analoghe a quelle svolte dalle componenti SVS e HASP.
Non abbiamo pertanto ritenuto significativo cambiare le dizioni OS e HASP all'interno del manuale, che e' stato invece aggiornato in altre parti fondamentali: le variazioni piu' rilevanti del JES2 rispetto ad HASP sono state riprese da un articolo di G.Mainetto e riportate in vari punti all'interno del manuale, mentre le variazioni dell'MVS rispetto all'SVS (e ulteriori funzioni aggiuntive valide nell'ambiente CNUCE dove opera sia un sistema OS, sia un sistema conversazionale VM) facevano parte di un articolo di A.Ceccarelli e sono state inserite integralmente come ultimo capitolo del manuale.

Informazioni complementari agli argomenti trattati in questo manuale possono essere reperite nelle seguenti pubblicazioni del CNUCE:

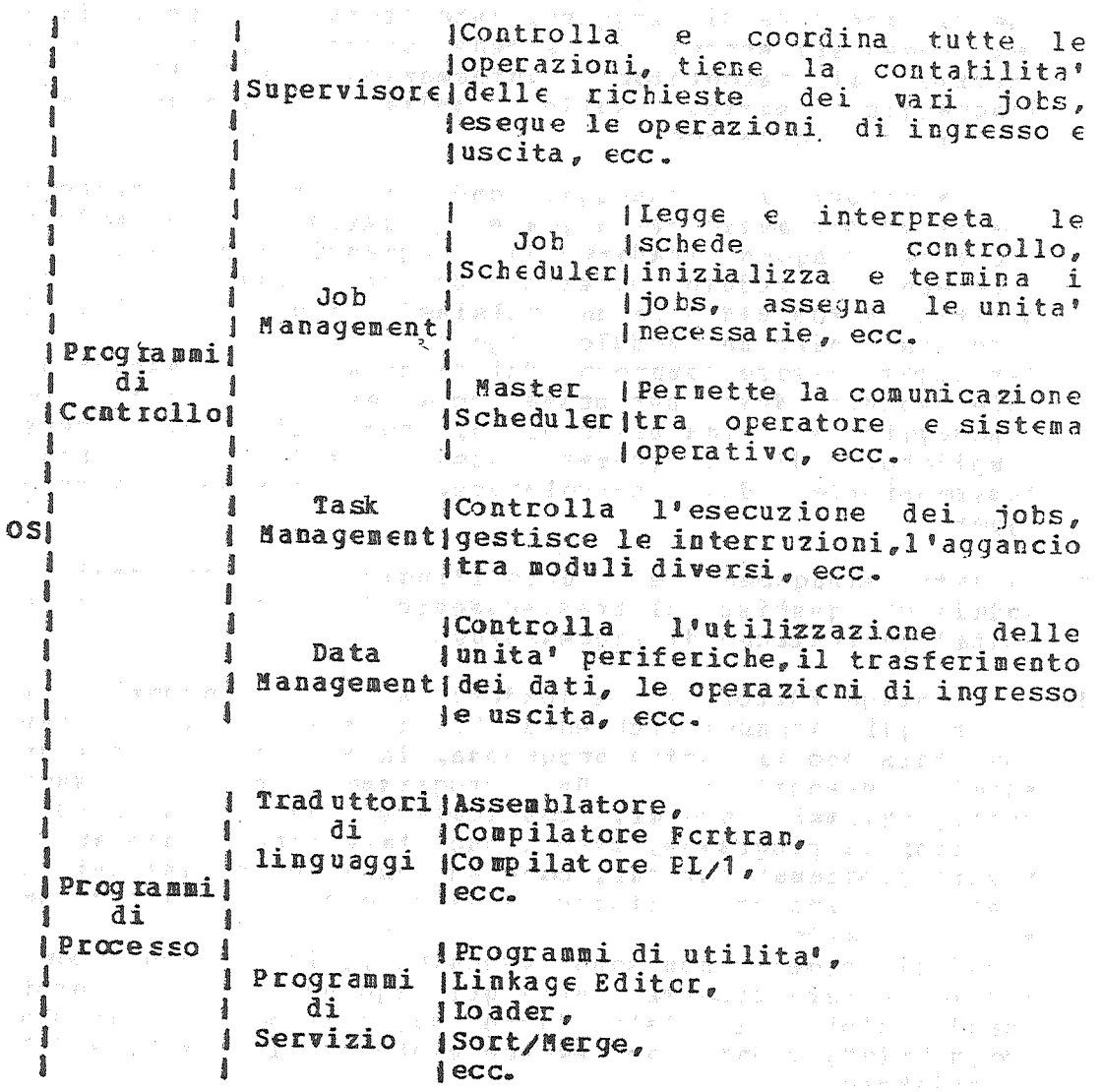
Pubblicazione	Argomento
ZC-50:	Programmazione Fortran sotto CS. Descrizione ed esempi delle schede controllo necessarie per la programmazione Fortran sotto CS; procedure catalogate relative, parametri, schede DD per l'accesso ai vari tipi di files trattati dal Fortran, esempi.
ZC-67:	Linkage Editor e Loader. Descrizione dei programmi di servizio Linkage Editor e Loader sotto il sistema operativo OS, delle possibilita' da essi offerte nella elaborazione dei moduli in ingresso, delle schede controllo e parole chiave necessarie ad effettuare le elaborazioni dovute in uscita ed esempi relativi.

1 - IL SISTEMA OPERATIVO CS

1a) Generalita'

Uno dei sistemi operativi che gestisce i calcolatori della serie IBM/360 e IBM/370 e' il sistema CS.
Esso si incarica di controllare tutta l'attivita' dei lavori che vengono immessi ed eseguiti su tali calcolatori, e di fornire loro tutte quelle risorse di cui necessitano, come ad esempio unita' di lettura e scrittura, il tempo necessario all'esecuzione del programma, la zona di memoria centrale richiesta, ecc.
Tutte queste informazioni sono fornite al sistema operativo per mezzo di schede, perforate secondo un tracciato opportuno, dette schede controllo. Le schede controllo costituiscono pertanto il mezzo di comunicazione tra il programmatore e il sistema operativo.
Il sistema operativo CS e' costituito da una serie di programmi di utilita', ciascuno dei quali assolve ad un determinato compito.
Insieme al sistema operativo OS, sul calcolatore opera anche un sottosistema, HASP, che permette la gestione altamente ottimizzata di tutte le unita' "unit record" (lettori, perforatori, stampatrici, ecc.) e dei terminali remoti connessi al sistema.
Come vedremo, il programmatore ha a sua disposizione oltre le schede controllo per l'OS (schede //) anche delle particolari schede controllo indirizzate esplicitamente verso il sistema HASP (schede /*) per poter ottenere particolari servizi.

Schematicamente il sistema operativo OS/HASP puo' essere cosi' rappresentato:



Di tutti i componenti presenti nello schema, quelli che piu' interessano il programmatore sono:

a) il Job Scheduler: e' quella parte del sistema operativo che si incarica di leggere, interpretare e controllare per eventuali errori le schede control, che devono fornire al calcolatore informazioni sul tipo di linguaggio usato, sulle unita' richieste per l'ingresso/uscita ecc.

b) i Traduttori di linguaggi: sono la parte del sistema operativo per mezzo della quale le istruzioni del modulo sorgente vengono tradotte in linguaggio-macchina, per ottenere il modulo oggetto. Un programma, sia esso scritto in FORTRAN o in un qualsiasi altro linguaggio, si dice costituire un "modulo sorgente".
Per poter essere compreso dal calcolatore ed eseguito, tale modulo deve per prima cosa essere tradotto in linguaggio macchina binario, o, come si dice, essere compilato. Il programma cosi' tradotto e reso comprensibile dal calcolatore, si chiama "modulo oggetto".

c) il Data Management: e' tutto l'insieme di programmi e moduli che gestisce il trasferimento dei dati da e per le unita' periferiche di ingresso/uscita.

d) il Linkage Editor e il Loader: sono i programmi che creano gli agganci richiesti tra i moduli oggetto che costituiscono il nostro programma, in modo da ottenere un modulo eseguibile. Un programma con i suoi sottoprogrammi interni, costituisce un unico modulo oggetto; il programma, pero', puo' fare riferimento anche a sottoprogrammi esterni, come ad esempio le funzioni di libreria: ciascun sottoprogramma esterno costituisce un modulo a parte.
I moduli oggetto non sono eseguibili, in quanto devono ancora essere risolti tutti gli agganci tra di loro: quando cio' sia stato fatto si ottiene un "modulo eseguibile", pronto per essere elaborato in memoria dal calcolatore.
Il Linkage Editor, a differenza del Loader, crea una versione del modulo eseguibile su disco ("modulo caricabile"): cio' permette di conservare su disco, in una propria libreria, un programma in forma direttamente eseguibile, senza la necessita' di ripetere ogni volta tutti i passi precedenti di compilazione e risoluzione degli agganci.
Cio' offre inoltre la possibilita' di creare delle strutture particolari nel modulo caricabile, risistemando in altro ordine le varie unita' che lo compongono: in particolare e' possibile creare delle strutture di "overlay" in tutti quei casi in cui l'occupazione di memoria di un singolo programma si rivelasse eccessiva.

1b) Principali differenze tra BATCH OS e CMS

a) Possibilita' di definire procedure catalogate speciali:

Come sara' meglio specificato nel seguito una procedura catalogata e' costituita da un insieme di schede EXEC e schede DD che servono a definire l'esecuzione di un determinato programma e le risorse di cui esso dovra' disporre: in tal modo per eseguire il programma richiesto sara' solo necessario richiamare quella data procedura con un'unica scheda EXEC; pensera' il supervisore a prelevare dalla opportuna libreria la procedura catalogata richiesta, ad iniziare l'esecuzione del programma in essa specificato e a fornire a questo tutte le risorse di cui necessita, tramite le schede DD definite nella procedura stessa.
Per incontrare le esigenze di ogni Utente, le schede DD che competono ad una procedura, possono essere sostituite, all'atto dell'uso della medesima, da altre schede DD definite dallo stesso Utente, senza per questo modificare la copia originale nella libreria delle procedure ("override" delle schede).

b) Presenza del Linkage Editor:

Un programma sorgente, scritto cioe' in un linguaggio simbolico (Fortran, PL/1, Assembler, ecc.) ha bisogno, come primo passo, di essere tradotto in linguaggio macchina.
Si occupano di cio' i cosiddetti traduttori di linguaggi o compilatori: ogni subroutine del programma sorgente viene cioe' tradotta in un modulo oggetto.
Un traduttore di linguaggio non risolve pero' tutti gli indirizzi o tutti i riferimenti simbolici contenuti nel modulo e pertanto i moduli oggetto creati non sono ancora eseguibili.
Essi devono essere elaborati da un programma di servizio che risolva questi riferimenti e crei un modulo in forma adatta per il caricamento in memoria e l'esecuzione: Linkage Editor e Loader sono due di tali programmi di servizio.
La sola differenza nell'uso del Linkage Editor rispetto al programma di Loader consiste nel fatto che il modulo generato viene memorizzato su disco prima di essere caricato in memoria per l'esecuzione: cio' permette di conservare su disco in una propria libreria un programma in forma direttamente eseguibile, risparmiando cosi' ogni volta il tempo di compilazione e riscluzione degli aggianci. Cio' offre inoltre la possibilita' di creare delle strutture overlay di moduli in forma eseguibile, in tutti quei casi in cui l'occupazione di memoria di un singolo programma si rivelasse eccessiva.
Osserviamo anche come il Linkage Editor accetti in input non solo moduli oggetto, ma anche moduli in forma caricabile prodotti in passi precedenti o facenti parte di librerie preesistenti.
Notiamo infine come la presenza del Linkage Editor permetta di implementare il concetto di tasks paralleli

proprio dell'OS.
Si intende con cio' la possibilita' di creare due moduli eseguibili mediante due applicazioni del Linkage Editor e di richiamare uno dei due dall'altro mediante l'uso della macro ATTACH: si ottiene in tal modo come risultato la esecuzione contemporanea dei due moduli, i quali sfruttano l'uno il tempo di canale dell'altro e viceversa, riducendo in tal modo il tempo di esecuzione totale di un lavoro.

c) Organizzazioni dei records particolarmente flessibili:

Il sistema operativo OS presenta in piu' rispetto al CMS le seguenti organizzazioni dei records all'interno del data sets a cui si riferiscono:

- organizzazione "partitioned": essa permette in modo semplice ed efficiente la gestione di piu' insiemi di dati sequenziali su unita' periferiche ad accesso diretto.
  Un data set partitioned (o "libreria") si compone cioe' di piu' gruppi di dati indipendenti chiamati "membri", ciascuno dei quali e' costituito da records organizzati sequenzialmente ed e' identificato da un nome (nome del membro).
  I nomi dei membri di un data set partitioned sono contenuti in un indirizzario ("directory"): ciascun nome e' accompagnato dall'indirizzo di partenza del membro e da altre informazioni sul membro stesso, cosi' che la semplice indicazione, da parte del programmatore, del membro e della libreria a cui tale membro appartiene permette al sistema di rintracciare rapidamente l'insieme cercato.

- organizzazione "index sequential": tale organizzazione permette un accesso rapido ed efficiente ad un singolo record del data set, facendo in modo che la posizione da esso occupata all'interno del data set stesso dipenda dal valore di una chiave di ricerca che fa parte del record medesimo.
  La creazione di un data set index sequential avviene mediante l'ordinamento dei records in base alla chiave: i records cosi' ordinati vengono registrati su un dispositivo ad accesso diretto. Ogni volta che una traccia viene riempita, la chiave piu' alta che ha trovato posto nella traccia viene memorizzata in una tabella di "indici di traccia" (track index). Quando tutte le tracce di un cilindro sono state riempite, la chiave piu' alta registrata nell'ambito del cilindro viene memorizzata in una tabella di "indici di cilindro" (cylinder index). Per leggere un record e' necessario fornire la chiave: da questa le routines del data management, valendosi della tabella degli indici, risalgono rapidamente all'indirizzo della traccia che contiene il record cercato. Il passo successivo consiste nella ricerca, all'interno della traccia, del record che contiene la chiave indicata.
  Questa organizzazione permette inoltre, mediante la

gestione di "aree di overflow", l'inserimento di nuovi records (o la modifica di records preesistenti) senza la necessita' di riscrivere l'intero data set. Essa consente infine l'elaborazione sequenziale e ad accesso diretto dei propri records.

- organizzazione "direct": con una tale organizzazione, per leggere o scrivere un record fisico (blocco) e' necessario fornire o l'indirizzo assoluto (cioe' il dispositivo di I/O, il cilindro, la traccia, la posizione del blocco) o l'indirizzo relativo all'inizio del data set.
  Se il blocco contiene una chiave, le routines del data management ricercano il blocco con la chiave specificata: la ricerca inizia all'indirizzo (assoluto o relativo) che e' stato fornito. La organizzazione diretta e' permessa solo su dispositivi ad accesso diretto.

d) Possibilita' di mantenimento di librerie su disco:

Il sistema OS permette di creare, all'interno di un job, dei data sets su disco e di conservarli e riutilizzarli successivamente qualora se ne presenti la necessita'.
In particolare possono essere conservate su disco le uscite del compilatore o del Linkage Editor: a scopo esemplificativo si consideri il caso in cui si voglia compilare ed eseguire un programma principale e le subroutines ad esso associate.
Valendosi della possibilita' suddetta, conviene memorizzare separatamente su disco i vari moduli compilati, per esempio quali membri di uno stesso data set partitioned, in modo che, se uno di essi presenta un errore in esecuzione o e' suscettibile di aggiornamenti, solo esso puo' venire modificato, ricompilato e reinserito nel data set al posto di quello errato, risparmiando cosi' sia il tempo di compilazione di tutti gli altri moduli, sia la necessita' di dover maneggiare le schede di tutto quanto il programma.

2- SCHEDE CONTROLLO OS/HASP

2a) Generalita'

Le schede controllo forniscono un mezzo di comunicazione tra il programmatore e i sistemi operativi OS e HASP: esse sono contraddistinte rispettivamente dalla coppia di caratteri // in colonna 1 e 2 e dalla coppia di caratteri /* sempre in colonna 1 e 2.
Nelle schede controllo // si distinguono i campi:

```
___________________________________________________
//Nome ¥ Codice operativo ¥ Operandi ¥ Commenti
|
|
```

I vari campi devono essere separati tra loro da almeno uno spazio bianco.

Nome : se presente, identifica lo statement di controllo per eventuali riferimenti. La sua lunghezza varia da 1 a 8 caratteri alfanumerici, di cui il primo deve essere alfabetico e deve iniziare necessariamente da colonna 3.

Codice operativo : specifica il tipo di statement di controllo. Puo' essere di tre tipi:

- JOB : individua l'inizio di un programma e comunica al sistema i parametri riguardanti il job e il programmatore, per usi statistici.
- EXEC : richiama in esecuzione una procedura o un modulo eseguibile che si trova in una libreria su disco.
- DD : (data definition) fornisce le indicazioni necessarie ad individuare i dati su cui lavora il modulo precedente e le unita' da esso utilizzate.

Operandi : contiene dei parametri informativi per il sistema operativo, separati tra di loro da virgole, e che verranno descritti nel seguito. Essi non devono mai contenere spazi bianchi, in quanto uno o piu' spazi bianchi indicano la fine del campo.
Quando il campo operando e' tale da superare la colonna 71 (*) occorre continuare in una o piu' schede successive. Per continuare il campo operando occorre:

- interrompere il campo dopo la scrittura completa di un parametro, inclusa la virgola che lo segue, non oltre la colonna 71 (*)
- opzionale: codificare un carattere non blank in colonna 72
- codificare i caratteri // nelle colonne 1 e 2 della scheda successiva
- continuare la sequenza dei parametri dell'operando a partire da una qualsiasi colonna compresa tra 4 e 16

Commenti : una sequenza di caratteri (comprendenti anche spazi biarchi) a scopo esplicativo: non vengono presi in considerazione del sistema operativo. La sequenza di caratteri non puo' superare la colonna 71.

Ricordiamo che un job e' costituito da uno o piu' passi di lavoro detti STEP: ciascuno STEP e' identificato da una scheda EXEC, che definisce il programma da eseguire, e da una o piu' schede DD che definiscono i data sets e le unita' utilizzati dal programma richiamato.
In genere quindi un job e' costituito come segue:

```
//nome JOB
//      EXEC     _
//      DD        |
//      DD        |
//      DD        |   STEP 1
        ....      |
        ....     _|
        _
//      EXEC     |
//      DD       |
//      DD       |   STEP 2
        ....     |
        ....    _|
        _
//      EXEC     |
//      DD       |
//      DD       |   STEP 3
        ....     |
        ....    _|
/*
```

(*) 70 per la scheda JOB (Vedi capitolo relativo)

2b) Scheda JOB

La scheda JOB e' la prima scheda del deck. Il formato della scheda JOB varia notevolmente da installazione a installazione. E' pertanto sempre necessario che il programmatore si informi sull'esatto tracciato di tale scheda presso il Centro nel quale intende eseguire i propri programmi.
Descriviamo il tracciato di tale scheda presso il CNUCE:

```
//nome JOB (---parametri1---),'---parametri2---',---parametri3---
```

dove

nome : da 1 a 8 caratteri alfanumerici, di cui il primo alfabetico, e' il nome che identifica il job;

parametri1 : e' una serie di informazioni, separate tra di loro da virgole e racchiuse nel loro insieme da una coppia di parentesi;

parametri2 : e' una serie di informazioni di formato variabile, e racchiuse nel loro insieme da una coppia di apici;

parametri3 : sono dei parametri in forma di parola.chiave, separati tra di loro da virgole e che non devono superare la colonna 70 della scheda; l'eventuale carattere di continuazione e' perforato in colonna 72.

---parametri---:
(persona,ente,tempo,linee,schede,carta,copie,log,pagina)

dove:

persona : (obbligatorio); quattro caratteri alfanumerici che identificano il presentatore del deck;

ente : (obbligatorio); quattro caratteri alfanumerici che identificano l'ente a cui appartiene il presentatore;

tempo : (opzionale,default=2); da una a quattro cifre numeriche indicanti, in minuti, il tempo di CPU richiesto dal job;

linee : (opzionale,default=1); da una a quattro cifre numeriche indicanti, in migliaia, le linee di stampa richieste dal jcb;

schede : (opzionale,default=100); da una a quattro cifre numeriche indicanti, in unita', le schede perforate richieste dal jcb;

carta : (opzionale,default=7111); quattro cifre numeriche indicanti, secondo le specifiche riportate a pagina seguente, la richiesta di catene, carta o nastrini particolari sulle stampatrici del calcolatore.

copie : (opzionale,default=1); da una a due cifre numeriche indicanti il numero di copie di output che si vogliono ottenere sulla stampatrice del calcolatore (max 30);

log : (opzionale,default=non-codificato); serve a includere o meno (codifica=N) le informazioni stampate all'inizio del jcb (joblog);

pagina : (opzionale,default=61); da una a due cifre numeriche indicanti il numero di linee per pagina che si vogliono ottenere in stampa. La codifica 0 (zero) consente di ottenere la stampa continua, senza salto pagina (max=255).

1a CIFRA: TIPO CATENA

	Catena numero	Tipo	UCS		Stampatrice
			VM	VS	
1	1	H11 normale	H11	H11	3211
2	2	T11 normale	T11	T11	3211
3	3 -	-	-	-	-
4	4 -	-	-	-	-
5	5 -	-	-	-	-
6	6	TN normale	TN	UN	1403
7	7	QN3 normale	QN	QN	1403
8	8 -	-	-	-	-
9	9 -	-	-	-	-
U	Catena consegnata dall'utente		-	-	-

2a CIFRA: TIPO CARTA

1	Carta 40x11 lettura facilitata (normale)
2	Carta 40x11 lettura facilitata (sul retro)
3	Carta 40x11 bianca
4	Etichette autoadesive
5	Carta lunga bianca
U	Carta consegnata dall'Utente

3a CIFRA: NASTRINO CONTROLLO

1	66 righe per pagina (normale)
2	88 righe per pagina
3	132 righe per pagina
U	Nastrino consegnato dall'Utente

4a CIFRA: DRIVE CONTROLLO

1	Carta 66 (normale)
2	Carta 88
3	Schede 8 inc
4	Schede 12 inc

I parametri sono nel loro insieme racchiusi da una coppia di parentesi, mentre sono separati uno dall'altro da una virgola.
Per omettere un determinato campo, basta codificare i due delimitatori che lo precedono e lo seguono (parentesi o virgole), contigui l'uno all'altro.
Se si vogliono omettere i parametri fino alla fine, non e' necessario codificare, in coda all'ultimo, le virgole rimanenti, ma e' sufficiente chiudere la parentesi dopo l'ultimo parametro esplicitamente codificato.

E' da tenere presente che la schedulazione dei jobs per l'esecuzione avviene da parte di HASP tenendo conto del tempo e delle linee richieste: minori sono questi parametri, tanto maggiore sara' la priorita' assegnata al job.
In esecuzione un controllo effettuato dal Supervisore sul tempo e sulle linee fa terminare l'esecuzione di un job che usi piu' del tempo richiesto o stampi piu' delle linee richieste.

---parametri2---:
'commenti'

commenti : (opzionali) una serie di caratteri alfanumerici o speciali (punti ecc.) atti ad identificare il job o il programmatore: il numero di caratteri codificabili e' variabile e dipende dalle lunghezze dei campi precedenti. In totale tra i due apici di apertura e chiusura di tutto il campo possono stare al massimo 20 caratteri.

---parametri3---:
REGION=regioneK,MSGLEVEL=(n,m),CLASS=x

dove:

REGION=regioneK : da una a quattro cifre numeriche indicanti in Kbytes (1K=1024 bytes), il valore della regione di memoria richiesta dal job. Il valore codificato viene automaticamente arrotondato dal sistema al multiplo di 64K immediatamente superiore.
Nel caso che tale parametro sia esplicitamente codificato sulla scheda JOB, il valore indicato verra' interpretato come regione di memoria virtuale richiesta per ciascuno degli steps di cui e' composto il job, qualunque sia il valore della REGION richiesta per i singoli steps. Nel caso che sulla JOB non compaia esplicitamente il parametro REGION, valgono i valori codificati per i singoli steps, esplicitamente o nell'eventuale procedura catalogata richiamata. Nel caso che manchi qualsiasi indicazione del parametro REGION, sia sulla JOB che sulle EXEC, il valore default e' di 128K.

MSGLEVEL=(n,m) : (opzionale,default=(2,1)); da' la possibilita' di ottenere o di sopprimere informazioni stampate sull'uscita del job, secondo la codifica:

n=0 viene listata, sull'uscita, la sola scheda JOB.

1 vengono listate sull'uscita tutte le schede controllo in ingresso, e tutta l'espansione delle eventuali procedure catalogate richiamate.

2 vengono listate sull'uscita tutte e sole le schede controllo in ingresso, senza l'espansione delle evertuali procedure catalogate richiamate.

m=0 viene soppressa, sull'uscita, la stampa dei messaggi di allocazione/disallocazione dei data sets utilizzati nel caso di fine normale del job. Nel caso di fine anormale la lista viene eseguita regolarmente.

1 vengono listati, sull'uscita, tutti i messaggi di allocazione/disallocazione dei data sets utilizzati nel job.

CLASS=x : mentre la priorita' di schedulazione di un job resta calcolata da HASP in base al tempo di CPU richiesto, ad un job viene assegnata automaticamente una determinata classe per l'esecuzione, in base alla richiesta di tempo di CPU e utilizzo di unita' periferiche, cosi' da ottimizzare nel modo migliore la gestione delle risorse da parte del sistema, oltre a permettere una schedulazione piu' accurata dei jobs, e quindi un lavoro piu' efficiente da parte dell'operatore. Se si vuol forzare un job ad essere eseguito nelle ore notturne a tariffa ridotta, va invece esplicitamente codificata la classe T.

2c) SCHEDE /*SETUP e /*MESSAGE

Sono le schede necessarie per la richiesta di dispositivi dedicati (nastri, dischi, stampatrici, ecc.) sul calcolatore. Esse devono essere inserite, nel deck, subito dopo la scheda JOB e prima di ogni altra scheda //.
I programmi che richiedono l'uso di dispositivi dedicati, devono presentare subito dopo la scheda JCB, le schede:

```
1 3                                                  7172  80
/*SETUP     b     argomenti ------                     b---b
/*MESSAGE   b     argomenti ------                     b---b
```

a) Gli argomenti delle schede SETUP indicano il numero di unita' richieste, tramite la codifica:

```
TPE9=n        per i nastri a 9 tracce e 800/1600    (n=1-3)
TPV9=n        equivalente a TPE9
TPH9=n        per i nastri a 9 tracce e 6250 Bpi    (n=1-4)
```

Le varie parole chiave vanno separate tra di loro da una virgola, possono essere codificate in ordine arbitrario e non devono superare la colonna 71.
Non sono ammessi caratteri blank nel campo degli argomenti.

b) Gli argomenti delle schede MESSAGE indicano il nome dei volumi che devono essere montati sulle unita' richieste tramite la codifica:

```
XXXX=(aaaaaa,bbbbbb,----------)
```

dove XXXX e' una delle parole chiavi presenti sulla scheda SETUP precedente e aaaaaa,bbbbbb, ecc. sono le label dei volumi (nastri, dispack, catene ecc.) che devono essere montati su tali unita'.

c) NASTRI RISERVATI (NO-LABEL e STANDARD-LABEL)

Ricordiamo che i nastri riservati in Sala Macchine vengono classificati in base ad una sigla composta da una lettera e tre numeri (es. X555, Y666 ecc.) che ne identifica univocamente la posizione negli appositi scaffali.

In genere uno di tali nastri viene richiesto in batch CS tramite la codifica di tale sigla sulla scheda DD e sulla scheda MESSAGE che si riferiscono ad esso.
Cosi' ad esempio la codifica:

```
/*SETUP        TPE9=1
/*MESSAGE      TPE9=(Y666)
......
//nome  DD  UNIT=TPE9,VOL=SER=Y666,...
```

causa la richiesta, sulla console dell'operatore, del montaggio del nastro Y666 su una unita' TPE9.

Alcune difficolta' sorgono se il nastro e' "Standard-Label": in tal caso infatti per il parametro VOL=SER=, e' obbligatoria la codifica della label effettiva del nastro (e non della sigla di riserva fornita dal CNUCE).
In tal caso pero' l'operatore non e' in grado di risalire, dalla label del nastro, alla sigla di classificazione, se non con le informazioni che l'utente stesso ha codificato sulla scheda MESSAGE: e' importante a questo proposito ricordare che le schede SETUP e MESSAGE vengono visualizzate a console solo nel momento in cui il job viene letto e che l'operatore puo', in un qualunque momento successivo, richiamare sullo schermo solo le informazioni codificate fino alla colonna 71 dell'ultima scheda MESSAGE presente nel job.

Si raccomanda pertanto di inserire informazioni importanti per la fase di esecuzione del job in tale campo: in particolare dovra' essere cura dell'Utente dare l'indicazione dei nastri richiesti e dell'eventuale corrispondenza tra la label di un nastro SL e la sigla di classificazione.
Cosi' mentre ad esempio la richiesta di due nastri (K888 e K889) senza label avveniva tramite le schede:

```
/*SETUP          TPE9=2
/*MESSAGE        TPE9=(K888,K889)
......
//DD1  DD  UNIT=TPE9,VOL=SER=K888,...
//DD2  DD  UNIT=TPE9,VOL=SER=K889,...
```

la richiesta di due nastri standard label, (X777 con label ZIPPO e X778 con label ZAPPO) dovra' avvenire con la codifica:

```
/*SETUP          TPE9=2
/*MESSAGE        TPE9=(X777-ZIPPO,X778-ZAPPO)
......
//DD1  DD  UNIT=TPE9,VOL=SER=ZIPPO,...
//DD2  DD  UNIT=TPE9,VOL=SER=ZAPPO,...
```

Si noti in particolare che, mentre il testo di <u>tutte</u> le schede SETUP e MESSAGE viene scritto sulla console dell'operatore durante la lettura del job, le informazioni codificate dalla colonna 16 alla colonna 71 dell'<u>ultima</u> scheda MESSAGE del job vengono visualizzate in ogni momento su richiesta dell'operatore.
Si consiglia pertanto di utilizzare il campo suddetto dell'ultima scheda MESSAGE per indicare i nomi dei volumi da montare ed altre informazioni utili (anello per i nastri, ecc.).
Inoltre le informazioni presenti sulle schede SETUP vengono riassunte sullo schermo in modo da notificare all'operatore, su richiesta dello stesso, il numero complessivo di unita' TPE9, o quello di unita' TPV9 richieste e l'indicazione se sono richiesti o meno altri tipi di urita'.

Si ricorda inoltre che le informazioni codificate sulla scheda SETUP servono ad effettuare una corretta suddivisione dei jobs da parte di HASP nelle varie classi di esecuzione associate agli iniziatori.
Si raccomanda pertanto di attenersi scrupolosamente alle indicazioni fornite in questo capitolo per quello che concerne la codifica delle schede, al fine di evitare che i jobs con richieste fatte in difformita' dalla norma vengano cancellati dal sistema o subiscano notevoli ritardi nell'esecuzione a causa di una errata schedulazione.

Esempi:

```
col.1
 |
 //nome     JOB  .....
 /*SETUP          TPE9=1,TPV9=2
 /*MESSAGE        TPE9=(X999),TPV9=(Y998,Y999)
 //         EXEC .....
 ......
```

```
col.1
 |
 //nome     JOB
 /*SETUP          TPE9=1
 /*MESSAGE        TPE9=(X999 CON ANELLO)
 //         EXEC .....
 .....
```

d) NASTRI PLOTTER

Per l'utilizzo dei nastri per il Plotter si deve fare uso delle schede SETUP E MESSAGE codificate come segue:

```
 /*SETUP          TPE9=1
 /*MESSAGE        TPE9=($PLOT)
```

e) NASTRI SCRATCH

Per l'utilizzo di un nastro scratch (disponibile cioe' solo per la durata del job) si deve fare use delle schede SETUP E MESSAGE codificate come segue:

```
 /*SETUP          TPE9=1
 /*MESSAGE        TPE9=(SCRTCH)
```

SCHEDULAZIONE DEI JOBS SOTTO OS/VS2-HASP

Se un job non ha specificato il parametro CLASS=T, allora la classe viene assegnata secondo i seguenti criteri:

Plotter	Nastri	CPU richiesta	Classe	Hold
NO	NO	≤3'	A	NO
		>3' e ≤10'	B	NO
		>10'	C	NO
	1 o 2 TPE9-TPV9	≤10'	D	NO
		>10'	E	NO
	1 TPH9	≤10'	F	NO
		>10'	H	SI
	altri	≤10'	G	SI
		>10'	H	SI
SI	qualsiasi	qualsiasi	K	NO

Se un job ha specificato la classe T (elaborazione notturna) allora si decide solo se esso debba essere messo in HOLD:

Plotter	Nastri	CPU richiesta	Classe	Hold
NO	<3	qualsiasi	T	NO
	≥3	qualsiasi	T	SI
SI	qualsiasi	qualsiasi	T	SI

Attribuzione della priorita' in ingresso

All'interno di una classe i jobs vengono eseguiti rispettando la loro priorita'. Questa e' stabilita prendendo in considerazione il tempo stimato di esecuzione: tanto piu' grande e' questo valore, tanto minore e' la priorita' assegnata al job.

Criteri per l'esecuzione

Un job che si trova sul disco di Spool, quando la lettura e' terminata, puo' essere eseguito se e' libero un iniziatore al quale sia associata la stessa classe del job.

Fra tutti i jobs che hanno la stessa classe e' scelto quello che ha la priorita' piu' alta, e tra quelli che hanno la stessa priorita', viene preferito quello che e' stato letto per primo.

Ad uno stesso iniziatore possono essere associate piu' classi diverse; in tal caso un iniziatore al quale siano ad esempio associate le classi FAC scegliera', per l'esecuzione, un job tra quelli di classe F. Solo se non vi sono jobs di quella classe, l'iniziatore scegliera' tra i jobs la cui classe e' definita per seconda (A nell'esempio), e cosi' via.

Inoltre la stessa classe puo' essere associata a piu' iniziatori diversi; quando piu' di un iniziatore ha possibilita' di dare inizio ad un job, viene attivato per primo l'iniziatore di numero d'ordine inferiore.

Il numero di iniziatori e l'associazione ad essi delle classi di esecuzione sono definiti dal personale che cura il sistema a seconda delle esigenze e dei dati di performance rilevati.
Per fare un esempio si puo' pensare ad una distribuzione del tipo:

N° d'ordine	Classi per cui e' abilitato
1	A
2	AB
3	BAKC
4	CBAJ
5	FEA
....	....

Attribuzione della priorita' in uscita

Quando un job e' terminato, gli viene attribuita una priorita' di uscita in base al numero di linee stampate e di schede perforate; i jobs vengono stampati e perforati rispettando questa priorita'. A parita' di priorita' sono preferiti i jobs che sono stati letti per primi.

2d) Scheda EXEC

E' la scheda mediante la quale si richiede al supervisore l'esecuzione di un certo programma o il richiamo di una certa procedura catalogata.
Il formato della scheda EXEC e' libero:

//nome EXEC programma,parametri ⊻ convertì

A colonna 1 e 2 devono essere perforati i due caratteri // .
Da colonna 3 puo' essere codificato un nome di riferimento (non obbligatorio), secondo le specifiche gia' fornite per la scheda JOB.
Segue la codifica EXEC che deve essere preceduta e seguita da almeno uno spazio bianco.
Segue la richiesta del programma voluto (per es. un Compilatore, il Loader, o un qualsiasi programma eseguibile caricato in una libereria su disco) mediante la codifica:

PGM=nome se si tratta di un modulo caricabile o eseguibile.
PROC=nome (o semplicemente nome) se si tratta di una procedura catalogata.

Esempio:

```
// EXEC ASMF
// EXEC FORTG
// EXEC PL1
```

per richiamare le procedure catalogate per l'esecuzione rispettivamente di un programma Assembler, Fortran e PL/1.

Il nome del programma e' seguito, eventualmente (separato da una virgola), dal campo dei parametri richiesti dal programma richiamato.
In genere tale campo e' costituito dalla parola chiave PARM seguita dalla lista dei parametri racchiusi da una coppia di apici o di parentesi e separati tra di loro da virgole.
Gli apici sono obbligatori se il campo contiene caratteri speciali (punti, ecc.).
Le parentesi sono obbligatorie se il campo e' spezzato su due schede: in tal caso i parametri che contengono segni speciali vanno singolarmente racchiusi tra apici:

```
PARM='1,2,3,4'            o  PARM=(1,2,3,4)
PARM=(1,2,
      3,4)                   (Obbligatorio)
PARM='P=1,Q=2,R=3,S=4'    o  PARM=('P=1','Q=2','R=3','S=4')
PARM=('P=1','Q=2',
      'R=3','S=4')           (Obbligatorio)
```

Il supervisore necessita, per eseguire un certo programma, di tutta una serie di informazioni che il programmatore deve fornire mediante appropriate schede controllo DD che seguono la scheda EXEC.
Per alleviare al programmatore la fatica di codificare ogni volta tali schede controllo, sono disponibili le procedure catalogate: esse costituiscono, ciascuna, un membro di una particolare libreria e contengono, oltre alla richiesta di esecuzione di un programma (// EXEC PGM=...) anche tutta la serie di schede DD, che definiscono le risorse delle quali tale programma potra' disporre.
Per il programmatore sara' sufficiente codificare:

// EXEC PROC=nome o semplicemente // EXEC nome

affinche' il supervisore provveda ad estrarre dalla libreria delle procedure tutte le informazioni necessarie alla esecuzione di tale programma.
E' lasciata al programmatore la possibilita' di modificare, per la durata del job, mediante l'uso di opportuni parametri da inserire nella scheda EXEC, le informazioni contenute nella libreria delle procedure catalogate.
E' opportuno notare che i parametri possono essere forniti a tre livelli distinti di procedimento:

- In fase di generazione del sistema operativo sono stati scelti quei parametri di uso piu' comune.
- I parametri immessi nella libreria delle procedure catalogate si sovrappongono a quelli definiti in generazione, se si fa uso delle procedure catalogate.
- Il programmatore puo', in ogni suo lavoro, modificare tali parametri in base alle sue esigenze.

Consideriamo come esempio una procedura catalogata dal tipo seguente:

```
Nome della procedura: PROVA
//   PROC  A=7,B=8
//STEP1 EXEC PGM=PROG1,PARM=('P1=5','P2=6','P3=&A','P4=&B')
   ....
```

Il programmatore richiamera' tale procedura con la scheda

```
//   EXEC PROVA
```

In tal caso egli informera' il supervisore di iniziare l'esecuzione del programma PROG1, passando a tale programma i parametri P1=5, P2=6, P3=7, P4=8.
Se ora il programmatore volesse eseguire il programma col valore P1=10, dovrebbe, come abbiamo detto, cambiare il

valore a tale parametro, codificandolo

```
//  EXEC PROVA,PARM.STEP1=('P1=10','P2=6','P3=7','P4=8')
```

Con tale codifica egli indicherebbe al supervisore di eseguire il programma PROG1, ma indicherebbe anche che il parametro PARM nella scheda EXEC identificata come STEP1, deve essere sostituito con quello fornito dal programmatore stesso.
Si faccia attenzione che con tale uso dell'override (sovrapposizione) e' necessario ricodificare tra apici tutti i parametri che comparivano nella procedura; nel caso se ne codificassero solo alcuni, per es. richiamando

```
// EXEC PROVA,PARM.STEP1=('P1=10')
```

il supervisore assumerebbe per gli altri parametri P2,P3,P4 i valori standard definiti in generazione, che potrebbero quindi anche essere diversi da quelli definiti originariamente nella procedura catalogata.
Per evitare quindi la codifica di tutti i parametri al programmatore, ogni volta che egli desideri variarne uno, i parametri piu' comuni sono stati messi in una forma particolare, detta "forma parametrica" (vedi P3 e P4 nell'esempio precedente).
Per variare uno di questi parametri, basta che il programmatore codifichi il suo solo valore nella scheda EXEC servendosi della opportuna parola chiave come segue:

```
// EXEC PROVA,A=12
```

Gli altri parametri restano con i valori che hanno nella procedura catalogata, e solo il valore del parametro P3 viene cambiato.

Esempio pratico:

si abbia la seguente procedura per la compilazione ed esecuzione di un programma Fortran (IGIFORT e' in questo caso il nome del compilatore Fortran)

Nome della procedura: FCRTG

```
//      PROC MAP=MAP,LIST=LIST
//COMP EXEC PGM=IGIFORT,PARM='&LIST,&MAP'
//...  DD   ...
 ....
```

e ammettiamo che in generazione in IGIFORT siano stati definiti i parametri NOLIST,NOMAP.

- richiamando semplicemente la procedura con
  // EXEC FORTG
  Il programmatore si trovera' a fare uso dei parametri LIST e MAP definiti nella procedura catalogata.

- se l'Utente desiderasse i parametri NOLIST,MAP e codificasse
  // EXEC FORTG,PARM.COMP='NOLIST'
  sbaglierebbe, in quanto il parametro PARM nella procedura catalogata verrebbe rimpiazzato completamente da quello fornito, e quindi non si avrebbe piu' l'opzione MAP, ma verrebbe presa quella fornita in generazione, che in questo caso e' NOMAP.

- perche' le cose vadano come desidera l'Utente, si dovrebbero rifornire tutti i parametri necessari:
  // EXEC FORTG,PARM.COMP='NOLIST,MAP'

- per evitare tale necessita', e' stata introdotta la forma parametrica: in questo caso per avere le opzioni NOLIST,MAP, non occorre fare l'override di tutto il parametro PARM, riscrivendo esplicitamente NOLIST,MAP (affinche', codificando solo NOLIST non venga presa l'opzione di generazione NOMAP, come gia' detto), ma basta richiamare la procedura con
  // EXEC FORTG,LIST=NOLIST
  perche' solo il parametro LIST venga cambiato nella procedura catalogata, mentre il parametro MAP rimanga invariato.

Come indicazione di una procedura catalogata tipo, diamo la lista schematizzata della procedura FORTGCIC, e descriviamone brevemente le caratteristiche salienti.

```
//FORTGCLC  PROC
//COMP      EXEC   PGM=IGIFORT
//SYSPRINT   DD    stampatrice
//SYSPUNCH   DD    perforatore
//SYSLIN     DD    modulo oggetto
//LKED      EXEC   PGM=IEWL
//SYSPRINT   DD    stampatrice
//SYSUT1     DD    area di lavoro
//SYSLIN     DD    modulo oggetto
//SYSLIB     DD    librerie esterne
//SYSLMOD    DD    modulo caricabile
//GO        EXEC   PGM=modulo eseguibile
//STEPLIB    DD    modulo caricabile
//FT05F001   DD    dati
//FT06F001   DD    stampatrice
//FT07F001   DD    perforatore
```

Come si vede dalla sequenza delle schede controllo, la procedura e' costituita da tre passi, contraddistinti ciascuno, nel loro ordine di esecuzione, dalle schede EXEC:

```
//COMP    EXEC    PGM=IGIFORT
```

richiama in esecuzione il compilatore Fortran G1, che, nel nostro caso ha il nome di IGIFORT.
Le schede DD che seguono, fino alla EXEC successiva contengono le informazioni per il Compilatore, e precisamente:

```
//SYSPRINT    DD    stampatrice
```

indica che tutti i messaggi che riguardano la compilazione del programma (messaggi di errore, indicazioni sulle variabili usate, lista del programma ecc.) dovranno essere inviati sulla stampatrice del calcolatore.

```
//SYSPUNCH    DD    perforatore
```

indica il perforatore di schede del calcolatore come unita' di uscita nel caso venga richiesto il deck binario del programma tradotto

```
//SYSLIN    DD    modulo oggetto
```

indica al compilatore una libreria su disco, dove dovra' essere messo il programma compilato (modulo oggetto). Una volta che il programma e' stato compilato, si passa al secondo passo, cioe' quello di risoluzione degli agganci con le librerie esterne:

```
//LKED    EXEC    PGM=IEWL
```

richiama in esecuzione il programma di servizio Linkage Editor, che ha il nome IEWL.
Tale scheda e' seguita come quella per il compilatore, da tutta una serie di schede DD contenenti informazione per il Linkage Editor:

```
//SYSPRINT    DD    stampatrice
```

indica che tutti i messaggi che riguardano la fase di Linkage Editor dovranno essere inviati sulla stampatrice del calcolatore

```
//SYSUT1    DD    area di lavoro
```

indica le aree di lavoro che il Linkage Editor puo'

utilizzare su disco

//SYSLIN DD modulo oggetto

indica la libreria nella quale il Compilatore aveva messo il modulo oggetto, e che ora deve essere elaborato per la risoluzione degli agganci con i sottoprogrammi esterni.

//SYSLIB DD librerie esterne

indica al Linkage Editor le fonti dalle quali puo' attingere per la ricerca dei sottoprogrammi esterni richiamati dal programma compilato in precedenza.

//SYSLMOD DD modulo caricabile

indica al Linkage Editor la libreria su disco nella quale dovra' essere messo il programma in forma caricabile.

Infine si passa al terzo passo, cioe' quello della effettiva esecuzione del programma:

//GO EXEC PGM=modulo eseguibile

richiama in esecuzione in memoria il programma compilato e link-editato precedentemente.

//STEPLIB DD modulo caricabile

indica al sistema operativo la libreria su disco nella quale si trova il programma in forma caricabile

//FT05F001 DD dati di ingresso

associa alla unita' logica 5 del Fortran usata in una istruzione del tipo READ (5,...) l'unita' del calcolatore che contiene i dati di ingresso per il nostro programma.

//FT06F001 DD stampatrice

associa alla unita' logica 6 del Fortran la stampatrice del calcolatore. Una istruzione del tipo WRITE (6,...) causa la stampa su carta dei dati indicati nella lista di uscita.

//FT07F001 DD perforatore

associa alla unita' logica 7 del Fortran il perforatore di schede del calcolatore. Una istruzione del tipo WRITE (7,...) causa la perforazione dei dati indicati sulla lista di uscita.

Dato che la stessa procedura catalogata cosi' inserita nella libreria su disco, sara' utilizzata da tutti i programmatori, in essa non sono inserite quelle schede che per loro natura, variano da programma a programma, e cioe':

- la scheda JOB, che dovra' essere fornita da ciascun programmatore con i propri valori di addebito.

- La scheda //SYSIN per il Compilatore, che indica dove il particolare programmatore ha posto il modulo sorgente che intende elaborare.

Inoltre delle schede DD per la fase di esecuzione sono inserite: la scheda FT07F001 che fa riferimento al perforatore, la scheda FT06F001 che fa riferimento alla stampatrice e la scheda FT05F001 col seguente formato:

```
//FT05F001   DD    DDNAME=SYSIN
```

cio' significa che la descrizione dei dati di ingresso e' demandata agli operandi di una ulteriore scheda DD, di nome SYSIN, che sara' fornita dal programmatore stesso.
Pertanto un programmatore che voglia richiamare la procedura FORTG, dovra' fornire:

- Una scheda JOB con i suoi parametri

- La scheda EXEC che richiama le procedure

- La scheda SYSIN che fa riferimento al modulo sorgente

- La scheda SYSIN che fa riferimento a eventuali dati da elaborare.

Tutte le schede fornite dal programmatore al momento in cui viene richiamata la procedura catalogata si aggiungono o si sostituiscono a quelle definite nella procedura catalogata stessa.
Descriviamo l'ordine con cui tali schede aggiuntive vanno fornite dal programmatore:

a) Vanno fornite prima tutte le schede che si riferiscono al passo di Compilazione, poi quelle che si riferiscono al passo di Linkage Editor e infine quelle per il passo di Esecuzione. Tali schede vanno opportunamente qualificate, affinche' possano essere assegnate alle dovute fasi di elaborazione: il nome della scheda DD va cioe' fatto precedere dal nome del passo (cioe' dalla scheda EXEC) nel quale vogliamo sia inserita tale DD. I due nomi saranno separati dal carattere punto. Cosi' ad

esempio la scheda SYSIN che definisce l'input per il compilatore (programma sorgente) sara' codificata come:

```
//COMP.SYSIN  DD
```

mentre la scheda SYSIN che fa riferimento ai dati che il programma dovra' leggere ed elaborare in fase di esecuzione sara' codificata come:

```
//GO.SYSIN  DD
```

b) Inoltre, nell'ambito di ciascun passo, vanno fornite prima le schede che gia' compaiono nella procedura catalogata e che noi vogliamo sostituire, poi quelle che non compaiono nella procedura e che noi vogliamo aggiungere.

c) Infine quelle che vogliamo sostituire vanno fornite nella stessa sequenza nella quale compaiono nella procedura catalogata.

- Sono stati unificati, nelle procedure di questo Centro, i nomi di tutti gli steps e precisamente:

  COMP : nome di tutti gli steps di compilazione.
  LKED : nome di tutti gli steps di linkage editor
  GO : nome di tutti gli steps di go e di loader.

  Pertanto in tutti i punti in cui, nei manuali, si fa riferimento a nomi tipo ASM, FORT, PL1L, si sostituisca COMP :

```
//ASM.SYSIN  DD *   --->   //COMP.SYSIN DD *
//FORT.SYSIN DD *   --->   //COMP.SYSIN DD *
//PL1L.SYSIN DD *   --->   //COMP.SYSIN DD *
```

- I parametri CONDC e CONDI danno i condition code nei passi di compilazione e linkage editor rispettivamente: il passo corrente non e' eseguito se il passo precedente presentava una condizione di errore maggiore di quella indicata da tali parametri.

- Si noti che le procedure catalogate di Fortran che contengono un passo di esecuzione prevedono la possibilita' di utilizzo delle seguenti unita' logiche senza la necessita' di definirle esplicitamente:

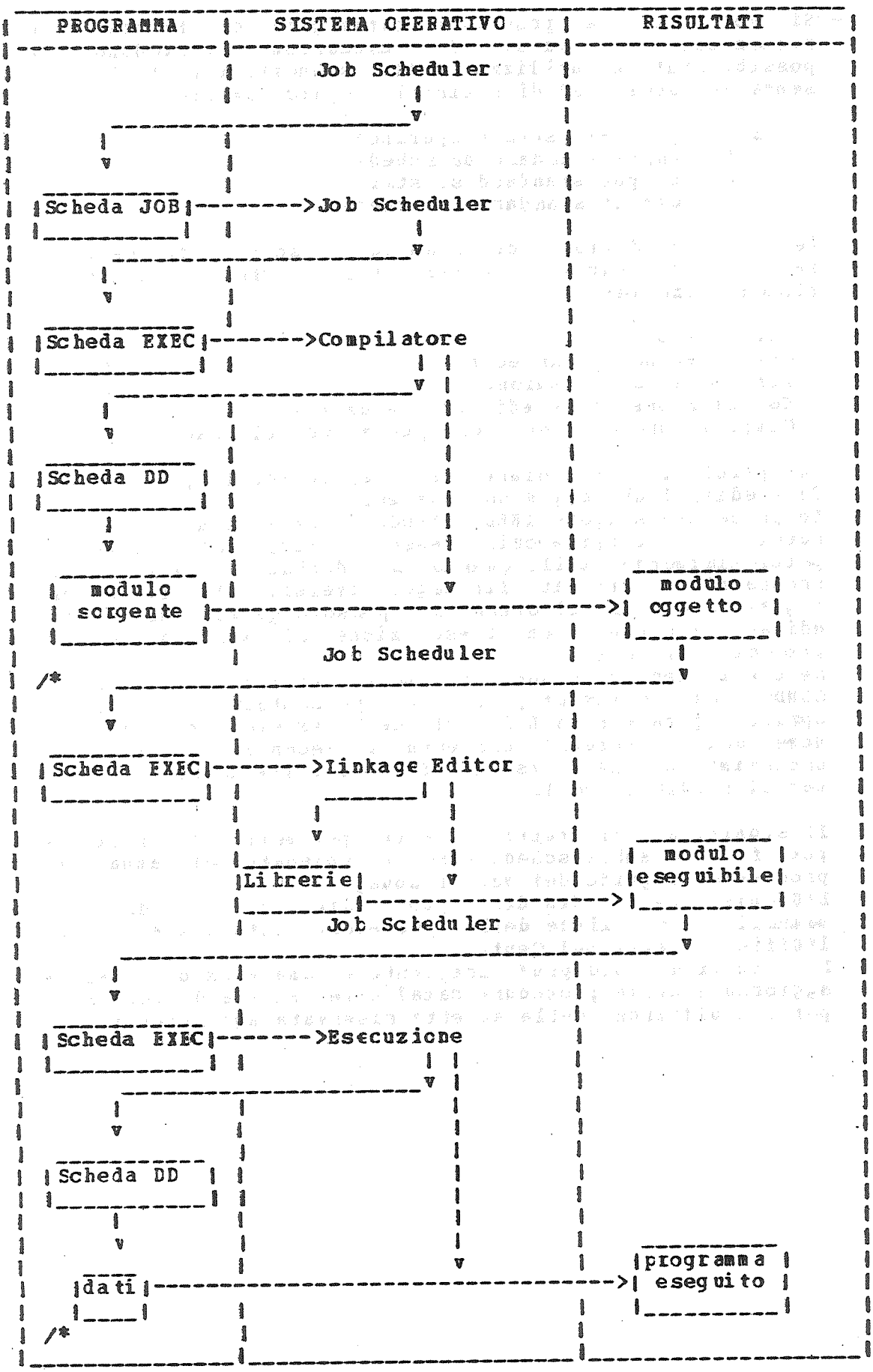
  1,2,3,4 : per data sets temporanei
  5 : input standard da schede
  6 : output standard su stampa
  7 : output standard su schede

- Per ogni traduttore di linguaggio sono a disposizione varie procedure catalogate, che consentono rispettivamente:

- Compilazione	C
- Compilazione e link edit	CI
- Link edit ed esecuzione	LG
- Compilazione, link edit ed esecuzione	CIG
- Compilazione ed esecuzione per mezzo del Loader	CG

- Due parole in particolare vanno spese per le procedure di link edit, link edit e go, loader.
  Le procedure singole IKED, IKEDGO e LOADER sono comuni per tutti i compilatori: esse sono, ad esempio, particolarmente utili quando si debbano compilare piu' programmi scritti in linguaggi diversi. I vari moduli oggetto che cosi' si ottengono possono quindi essere link editati insieme con l'esecuzione di una delle tre procedure elencate.
  Le tre procedure singole non hanno definito il parametro COND esplicitamente, mentre presentano l'ulteriore opzione parametrica LIB, che deve essere impostata col nome della libreria corretta a seconda del tipo di programma compilato (es: LIB=LINK per l'Assembler, LIB=FX per il Fortran, ecc.).

- Il significato di tutti gli altri parametri che l'Utente puo' fornire sulla scheda EXEC e' spiegato nei manuali di programmer's guide dei vari linguaggi.
  L'Utente che lo desidera puo' fare richiesta di tali manuali e di liste delle procedure catalogate presso l'Ufficio Utenti del Centro.
  I manuali di uso piu' frequente e una versione sempre aggiornata delle procedure catalogate sono a disposizione per consultazione nella saletta riservata agli Utenti.


PROGRAMMA
SISTEMA OPERATIVO
RISULTATI
Job Scheduler
Scheda JOB
Job Scheduler
Scheda EXEC
Compilatore
Scheda DD
modulo sorgente
modulo oggetto
Job Scheduler
/*
Scheda EXEC
Linkage Editor
Librerie
modulo eseguibile
Job Scheduler
Scheda EXEC
Esecuzione
Scheda DD
dati
programma eseguito
/*

2e) Scheda DD

Prima di passare a descrivere in dettaglio le schede controllo DD, e' necessario fornire i concetti di alcuni termini che si trovano nel sistema operativo OS.

Volume : un qualsiasi mezzo atto a contenere dei dati (ad es: schede, nastri, dischi, ecc.). Ciascun volume puo' essere identificato da un'etichetta (label) formata da una sequenza di caratteri alfanumerici (di cui il primo alfabetico) che ne definisce univocamente il nome.
Le schede non hanno una label, tutti i dischi hanno una label, i nastri possono o no avere una label.

Data set : un qualsiasi insieme di dati, a cui sia stato associato un nome (sequenza da 1 a 8 caratteri alfanumerici, di cui il primo alfabetico).
Un data set ha come attributi il formato e la organizzazione.

Formato di un data set : in generale una istruzione di scrittura, causa la creazione di un record logico o semplicemente "record": si chiama cioe' record l'insieme delle informazioni logiche che vengono elaborate nella singola istruzione di scrittura.
Piu' records logici possono essere uniti insieme per costituire quello che si dice un record fisico o "blocco": si parlera' allora di records bloccati.
I blocchi vengono trasferiti sul supporto fisico che li deve contenere e vengono separati tra di loro da un campo particolare (gap). Possiamo in base a cio' distinguere il formato del records in:

records a lunghezza fissa (RECFM=F):
1) tutti i records hanno la stessa lunghezza
2) se le informazioni trasferite hanno una lunghezza minore del record, questo viene riempito di spazi bianchi
3) se le informazioni trasferite hanno una lunghezza superiore al record si ha un troncamento e una segnalazione di errore
4) ciascun blocco e' costituito da un solo record

records a lunghezza fissa bloccati (RECFM=FB):
1,2,3) come il precedente

4) la lunghezza del blocco deve essere un multiplo della lunghezza del record

records a lunghezza variabile (RECFM=V):
1) i records hanno lunghezza diversa uno dall'altro. Ciascuno di essi porta un campo iniziale di 4 bytes con l'indicazione della lunghezza effettiva
2) se le informazioni trasferite hanno una lunghezza minore del massimo indicato, riempiono completamente un record logico e non si hanno bytes perduti
3) se le informazioni trasferite hanno una lunghezza superiore al massimo indicato, si ha un troncamento e una segnalazione di errore
4) ciascun blocco e' costituito da un solo record e porta in testa un campo di 4 bytes che contiene la lunghezza del blocco stesso

records a lunghezza variabile bloccati (RECFM=VB):
1,2,3) come il precedente
4) la lunghezza del blocco deve essere un multiplo della lunghezza del massimo record indicato piu' un campo di 4 bytes con l'indicazione della lunghezza del blocco stesso.

records variabili spanned (RECFM=VS):
1,2,3) come per i records a lunghezza variabile
4) la lunghezza del blocco e' indipendente dalla lunghezza del record: se infatti la lunghezza del record e' maggiore di quella del blocco, il record viene spezzato e ciascun segmento viene inserito in un blocco diverso. Ciascun blocco contiene un solo segmento di record

records variabili spanned bloccati (RECFM=VSB):
1,2,3) come il precedente
4) ciascun blocco puo' contenere piu' segmenti di record

records undefined (RECFM=U):
1) i records non portano caratteri di controllo in testa, come avveniva per i records a lunghezza variabile

2) se le informazioni trasferite hanno una lunghezza inferiore al massimo indicato, riempiono completamente un record e non si hanno bytes perduti
3) se le informazioni trasferite hanno una lunghezza superiore al massimo indicato, si ha un troncamento e una segnalazione di errore
4) ciascun blocco e' costituito da un solo record

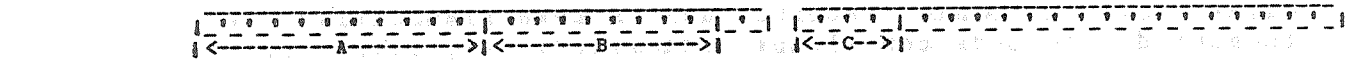
Diamo nella seguente tabella uno schema riassuntivo dei formati dei records con alcune indicazioni comparative sulle prestazioni di ciascuno: nella simbologia le prestazioni scadono di qualita' nel passaggio ++,+,-,--:

	Perdita bytes per records	Perdita bytes causa carattere di controllo	Perdita bytes causa "gap"	Perdita bytes per traccia di volume	Tempo di I/O	Tempo di supervisore
F	-	+	-	-	-	+
FB	-	+	+	-	+	+
V	+	-	-	-	-	-
VB	+	-	+ +	-	+ +	-
VS	+	- -	- -	+	- -	- -
VSB	+	- -	+	+	+	- -
U	+	+	-	-	-	+

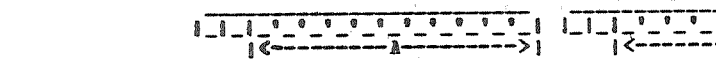
Esempio: supponiamo di dover trasferire tre records A, B e C rispettivamente di 44 bytes, 36 bytes e 16 bytes. Nelle figure che seguono, ciascun quadratino rappresenta una lunghezza di 4 bytes.

1) RECFM = F, LRECL = 44, BLKSIZE = 44

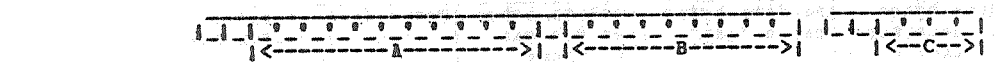
2) RECFM = FB, LRECL = 44, BLKSIZE = 88



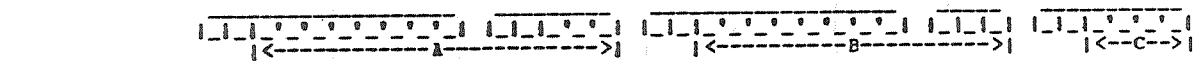
3) RECFM = V, LRECL = 48, BLKSIZE = 52



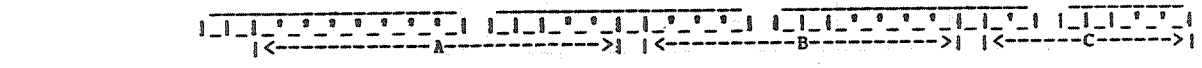
4) RECFM = VB, LRECL = 48, BLKSIZE = 100



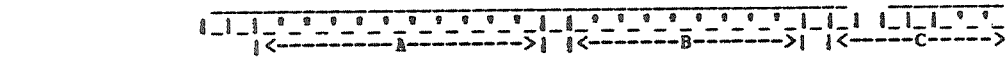
5) RECFM = VS, LRECL = 48, BLKSIZE = 40



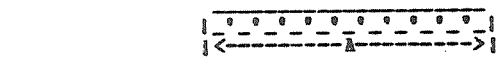
6) RECFM = VBS, LRECL = 48, BLKSIZE = 40



7) RECFM = VBS, LRECL = 48, BLKSIZE = 100



8) RECFM = U, BLKSIZE = 44



Organizzazione di un data set:

sequential data set (DSORG=PS):
insieme di dati identificati da un nome, ai quali si fa accesso partendo dall'inizio ed elaborandoli nella sequenza fisica con cui sono scritti sul volume, uno dopo l'altro (es: schede, nastri ecc.).

partitioned data set o libreria (DSORG=PO):
e' composto da piu' insiemi di dati, ciascuno dei quali e' organizzato sequenzialmente, ha un proprio nome, si chiama membro del data set partitioned ed e' indipendente dagli altri. L'elaborazione di uno di tali insiemi avviene fornendo il nome della libreria e il nome del membro all'interno della libreria stessa. Questa organizzazione puo' ad esempio essere utile per creare una libreria di sottoprogrammi, in cui cioe' ciascun membro del data set sia una subroutine e si possa pertanto accedere alla libreria per modificare ed elaborare i sottoprogrammi esistenti, o aggiungerne altri. Ad ogni data set partitioned e' associato un indirizzario (directory), che contiene l'elenco dei membri che tale data set comprende.

direct-access data set (DSORG=DA):
e' costituito da un insieme di records organizzati in maniera sequenziale, ma ciascuno dei quali e' identificato dalla sua posizione a partire dall'inizio del data set. E' pertanto possibile elaborare, all'interno del data set, direttamente un certo record, senza avere la necessita' di leggere tutti i precedenti come avveniva per i data sets sequenziali.

index-sequential data set (DSORG=IS):
e' costituito da un insieme di records organizzati in maniera sequenziale, ma ciascuno dei quali contiene una "parola chiave" che lo identifica univocamente. E' pertanto possibile elaborare, all'interno del data set, un particolare record voluto indicando la parola chiave che lo identifica.

Si desidera ricordare in questo contesto quanto segue, come descritto nei manuali Fortran Language e Fortran programmer's guide.

1) Le operazioni di I/O con FORMAT su data sets sequenziali accettano formati dei records di qualunque tipo (F,V,U ecc.);

```
es:      WRITE  (1,3) A,B,C
       3 FORMAT (----)
```

2) Le operazioni di I/O senza FORMAT su data sets sequenziali accettano i soli formati dei records VS e VSB;

```
es:      WRITE  (1) A,B,C
```

3) le operazioni di I/O con FORMAT su data sets ad accesso diretto accettano il solo formato dei records F;

```
es:      DEFINE FILE (----)
         WRITE  (1'N,3) A,B,C
       3 FORMAT (----)
```

4) le operazioni di I/O senza FORMAT su data sets ad accesso diretto accettano il solo formato dei records F;

```
es:      DEFINE FILE (----)
         WRITE  (1'N) A,B,C
```

Si rende noto a tale proposito che le procedure catalogate Fortran del Centro hanno per i data sets temporanei 1,2,3 e 4 (schede DD di tipo FT01F001 ecc.) il formato dei records VS.
L'Utente che desiderasse un formato diverso dovra' provvedere a fare l'override di tali schede.

Passiamo ora a parlare esplicitamente delle schede DD.

Esse sono schede mediante le quali il programmatore definisce le caratteristiche dei data sets che vuole usare. Tali informazioni si possono dividere in due classi:

- informazioni che caratterizzano il data set indipendentemente dal supporto fisico su cui esso risiede.
- informazioni che riguardano il supporto fisico su cui il data set risiede.

Il tracciato della scheda DD e' libero:

//ddname DD parametri /* commenti

A colonna 1 e 2 devono essere perforati i due caratteri // . Da colonna 3 e' in genere codificato il nome della scheda DD (ddname) secondo le specifiche gia' fornite per la scheda JOB.
Il ddname e' in genere collegato col nome del file che viene trattato all'interno del programma (si vedano per questo i programmer's guide dei vari linguaggi e gli esempi alla fine di questo manuale). E' tuttavia utile dare qui di seguito una lista di ddname usati dai traduttori di linguaggi e programmi di servizio, schematizzandone il significato:

ddname	funzione	parametri o nome del data set usato
SYSIN	programma sorgente	*
SYSPRINT	listings,errori,dumps	SYSOUT=A
SYSPUNCH	punch del deck binario	SYSOUT=B
SYSUT1	data sets	-
SYSUT2	di lavoro	-
SYSUT3	per i	-
SYSUT4	compilatori	-
SYSLIN(*)	output del compilatore e input al Lingage Editor e al Loader	&&LOADSET
SYSLMOD	output del Linkage Editor	&&GOSET(GO)
SYSLOUT	messaggi del Loader	SYSOUT=A
SYSLIB	librerie opzionali	-
SYSABEND	dump completo di memoria	SYSOUT=A
SYSUDUMP	dump della regione	SYSOUT=A

(*) L'Assembler-VS/VM fa eccezione facendo riferimento, in fase di compilazione, ad una scheda DD di nome SYSGO. L'Assembler-E segue invece la regola generale.

Sulla scheda segue poi la codifica DD, preceduta e seguita da almeno uno spazio bianco, e successivamente le informazioni sul data set, separate da virgole:

DSN o DSNAME : nome del data set, da 1 a 8 caratteri alfanumerici, il primo alfabetico. Se il data set deve servire solo per la durata del job, basta far precedere il nome dai due caratteri && che definiscono appunto un data set temporaneo. Se si vuol far riferimento a un membro di un data set partitioned, il nome del membro viene messo fra parentesi dopo quello del data set a cui si fa riferimento, senza lasciare alcun spazio bianco fra di essi:
DSN=dsname(member)

DISP=(p1,p2,p3): serve a fornire indicazioni sullo stato del data set.

p1: definisce lo stato attuale del data set.

NEW: se il data set deve essere creato e non esisteva prima.

OLD: se il data set era gia' esistente sul device e vogliamo accedervi dall'inizio.

MOD: se il data set era gia' esistente sul device e vogliamo posizionarci alla fine di esso per aggiungervi altre informazioni. Su disco, nel caso che il data set non esista, la specifica MOD equivale alla specifica NEW.

SHR: se il data set deve essere reso disponibile contemporaneamente a piu' Utenti. Se il data set e' una libreria comune e' obbligatorio definirlo shared.

p2: definisce come deve essere lo stato del data set alla fine dello step, ammesso che questo termini normalmente.

KEEP: se vogliamo che il data set venga conservato.

PASS: se vogliamo che il data set venga passato allo step successivo, e venga cosi' ritrovato senza che sia necessario fornire le informazioni sulla UNIT, VOL e LABEL.

DELETE: se vogliamo che il data set venga distrutto.

p3: definisce come deve essere lo stato del data set alla fine dello step, se questo termina in modo anormale.

KEEP: se vogliamo che il data set venga conservato.

DELETE: se vogliamo che il data set venga distrutto.

UNIT= : definisce l'unita' su cui e' o deve essere posto il data set in questione.
TPE7 : nastri a 7 piste e 200-556-800 Bpi di densita', con translator e converter (1 unita')
TPE9 o TPV9 : nastri a 9 piste e 800-1600 Bpi di densita' (9 unita')
TPE9 : nastri a 9 piste e 1600-6250 Bpi di densita' (2 unita')
SYSDA: dischi

VOL=SER= : definisce il numero di serie del volume che e' montato sull'unita' specificata e che deve contenere il data set: ad es. WORK01 o WORK02 per i dischi di lavoro temporaneo, una label per i nastri, ecc.

SPACE=(blk,(n,m,k)): definisce lo spazio che deve essere riservato su DASD ad un data set nuovo (al momento cioe' della sua creazione). Non va codificato per un data set gia' esistente. Tale parametro non va codificato per un data set che deve risiedere su nastro.
blk: rappresenta l'unita' nella quale viene allocato lo spazio primario.
TRK: lo spazio viene allocato in tracce (ricordiamo che una traccia di 3330 contiene circa 13030 bytes, una di 3350 circa 19069).
CYL: lo spazio viene allocato in cilindri (ricordiamo che per un 3330 un cilindro e' composto da 19 tracce, per un 3350 da 30).
num: un numero decimale indicante il numero di bytes che vogliamo venga acquisito come allocazione primaria per il data set.
n: numero di tracce o cilindri o blocchi che devono essere assegnate al data set.
m: se lo spazio definito (n tracce o cilindri o blocchi) non risultasse sufficiente, vengono allocate m tracce o cilindri e cosi' via, per un massimo di 15 volte.
k: e' codificato per un data set partitioned e fornisce il numero di blocchi di 256 bytes allocati per il directory del data set (si tenga conto che in media ciascun blocco contiene circa 5÷10 entrate).

LABEL=(n,label) : viene utilizzato per i nastri: n indica il numero progressivo del file sul nastro (se e' uno puo' essere omesso); label indica se il nastro ha standard label (n,SI), se non ha label (n,NL) o se non deve essere effettuato alcun controllo sulla label (n,BLP).

DCB=(RECFM=formato,LRECL=length,BLKSIZE=block,DEN=densita',
TRTCH=codifica)
formato: definisce il formato dei records di cui e' composto il data set:
F :fixed
FB :fixed blocked
V :variable
VB :variable blocked
VS :variable spanned
VSB :variable spanned blocked
U :undefined
length : lunghezza in bytes del record (record logico)
block : lunghezza in bytes del blocco (record fisico)
densita': usabile solo con i nastri, ne definisce la densita' :
0: 200 Bpi
1: 556 Bpi
2: 800 Bpi
3: 1600 Bpi
4: 6250 Bpi
codifica: usabile solo con i nastri a 7 piste definisce il modo con cui i caratteri di 6 bits devono essere trattati durante il trasporto da nastro a memoria (caratteri di 8 bits) e viceversa:
non codificato o E: (a seconda che la parita' sia dispari o pari). In scrittura si perdono i primi due bits di ogni byte, in lettura i primi due bits vengono messi a zero.
T o ET: (a seconda che la parita' sia dispari o pari). In scrittura viene fatta la conversione EBCDIC-ECL, in lettura viene fatta la conversione inversa.
C: (solo con la parita' dispari). In scrittura vengono messi 3 bytes consecutivi in memoria (3*8=24 bits)in 4 tracce su nastro (4*6=24 bits), in lettura viene fatta la conversione inversa.

Parita'	Conversion ON	Translation CN	Conv. & Trans. OFF
Pari	Non possibile	TRTCH=ET	TRTCH=E
Dispari	TRTCH=C	TRTCH=T	TRTCH non codificato

Altri operandi caratteristici della scheda DD sono:

* : indica che il data set definito e' quello che segue tale scheda fino a che non si trova una scheda /* o // che indica la fine del data set.

DATA : come * con la sola differenza che la fine del data set e' indicata solo dalla scheda /*, cioe' il data set puo' contenere anche schede che inizino con //.

Le schede DD * (e DD DATA) permettono l'uso di un proprio delimitatore di 2 caratteri al posto della scheda standard // o /*: cio' permette di poter inserire tra i dati anche le schede // e /*
es:

```
//SYSIN   DD   *,DLM=$$
--- dati comprendenti anche // e /* ----
$$
```

L'uso di caratteri speciali quali delimitatori comporta la codifica tra apici, nel parametro DLM, del delimitatore stesso (Es. DLM='::').
Il carattere &, avendo particolari significati, per poter essere indicato nel parametro DLM quale delimitatore, deve essere raddoppiato (Es. DLM='&&&&').

SYSOUT=A o B : indica la classe di output a cui deve essere indirizzato il data set:
A : il data set deve essere inviato sulla stampatrice
B : il data set deve essere inviato sul perforatore

DDNAME=ddname : indica che le informazioni richieste si trovano su una scheda DD, identificata col nome ddname.

Note varie:

- Sono permanentemente montati dispack di 3350 per le allocazioni temporanee di data sets della capacita' massima di circa 300 milioni di bytes ciascuno (WORK01, WORK02, ecc). A seconda delle esigenze il numero di tali dispack variera' nel corso della seduta operativa. E' pertanto opportuno richiedere l'uso di tali dischi di lavoro tramite il parametro UNIT=SYSDA, senza far riferimento al nome esplicito del disco (parametro VOL omesso), lasciando al sistema operativo il compito di ricercare su uno qualsiasi di essi lo spazio richiesto.
  I data sets allocati su tali dischi hanno in genere la durata del job e non viene garantita la loro sussistenza al di fuori di tale limite di tempo.
  Se invece si desidera far uso di un particolare disco, l'Utente dovra' codificare oltre al DSN, DISP, e UNIT=SYSDA, anche il parametro VOL=SER=label, che indichera' la label del disco che si intende utilizzare per i propri lavori.

- Perche' l'Utente possa conservare dei suoi data sets per periodi di tempo piu' lunghi sono messe a disposizione delle aree di libreria sulle quali l'Utente puo' memorizzare i suoi dati previo l'affitto di un determinato numero di tracce o cilindri, secondo le procedure stabilite dal Centro con separate comunicazioni.

- Un data set con una DISP=(NEW,KEEP) viene ritrovato, in uno step successivo, mediante la codifica dei parametri DSN, DISP, UNIT=SYSDA e VOL.

- Un data set con una DISP=(NEW,PASS) viene ritrovato, in uno step successivo, mediante la codifica dei soli parametri DSN e DISP.

- Un data set di sistema (SYS1.FORTLIB, ecc.) deve sempre essere referenziato mediante la codifica dei soli parametri DSN=nome e DISP=SHR.

- Se la codifica dello spazio primario e' stata effettuata in TRK o CYL, anche lo spazio secondario verra' allocato su tale base:

  SPACE=(TRK,(3,2)),DCB=(BLKSIZE=3000,....)

  vengono allocate tre tracce come spazio primario e, quando necessario, altre 2 tracce per volta fino ad un massimo di 15 volte.

- Se invece la codifica dello spazio primario e' stata

effettuata in bytes, le eventuali allocazioni secondarie faranno riferimento al valore specificato nel sottoparametro BLKSIZE del parametro DCB:

```
SPACE=(5200,(3,2)),DCB=(BLKSIZE=3000,....)
```

vengono allocati 3 blocchi da 5200 bytes come spazio primario e, quando necessario, altri 2 blocchi da 3000 bytes per volta, fino ad un massimo di 15 volte.

- La codifica del parametro SPACE su una scheda DD che fa riferimento ad un data set gia' esisterte, puo' causare, nel caso che lo spazio primario non sia piu' sufficiente a contenere i dati, una estensione di tale data set a seconda di quanto specificato nel parametro stesso e indipendentemente da quanto era stato specificato nella allocazione primitiva per tale data set.

  Esempio: Data set creato con SPACE=(CYL,(2,1)),DISP=NEW
  Vengono riservati due cilindri al data set.
  In una fase successiva si fa riferimento a tale data set:

senza il parametro SPACE	con SPACE=(TRK,(4,3))
Se i due cilindri primari non sono piu' sufficienti a contenere i dati, il data set viene esteso di un cilindro alla volta e per un massimo di 15 volte.	Se i due cilindri primari non sono piu' sufficienti a contenere i dati, il data set viene esteso di 3 tracce alla volta e per un massimo di 15 volte. In tal caso il parametro che indica lo spazio primario (4 tracce) viene ignorato. E' tuttavia necessario, per poter operare in questo modo, che in fase di creazione primitiva del data set, sia stata indicata una allocazione secondaria.

- In particolare si faccia attenzione che, nell'utilizzo di una procedura catalogata di Compile and Link per inserire un proprio programma in una libreria privata esistente, la codifica della scheda:

```
//LKED.SYSLMOD  DD  DSN=MYLIB(MYPROG),DISP=OLD,
//              UNIT=SYSDA,VOL=SER=MYDISK
```

senza una codifica esplicita del parametro SPACE, comporta

l'estensione della libreria MYLIB a seconda delle specifiche del parametro SPACE contenuto nella procedura catalogata utilizzata (e non col valore fornito in fase di creazione del file).
Per evitare una estensione indebita, e' necessario annullare l'effetto del parametro SPACE della procedura, mediante la codifica

```
//LKED.SYSLMOD  DD  DSN=MYLIB(MYPROG),DISP=OLD,SPACE=,
//              UNIT=SYSDA,VOL=SER=MYDISK
```

- Sostituzione e aggiunta di parametri o di schede DD :
oltre a quanto gia' detto nel capitolo 2d sull'ordine in cui si devono fornire le schede DD che vogliamo sostituire o aggiungere in un determinato job, a quelle presenti nelle procedure catalogate, si desidera qui precisare che:

ciascuno dei parametri codificati (o sottoparametri del DCB) nella scheda DD fornita, va a sostituire il corrispondente parametro nella scheda DD sostituita, o vi si aggiunge, nel caso che tale parametro non compaia su tale scheda;

per poter annullare un parametro preesistente, basta fornire, sulla nuova scheda, la parola chiave indicante tale parametro seguita dal segno uguale e senza codificare esplicitamente nessun valore; se si devono fornire altri parametri, la virgola di separazione dovra' seguire immediatamente il segno uguale senza nessun carattere in mezzo;

per poter cambiare o annullare il parametro DCB, occorre fornire esplicitamente tutti i sottocampi che lo compongono (RECFM,LRECL,BLKSIZE, ecc.), ciascuno dei quali va a modificare il sottocampo corrispondente sulla scheda preesistente;

per poter modificare dei parametri di schede DD tra loro concatenate occorre fornire tante schede DD concatenate, quante necessarie a sovrapporsi alla scheda voluta, e codificare, nella DD corrispondente a tale scheda i cambiamenti voluti.

Esempio:
Una procedura catalogata contenga, nella fase Linkage Editor le seguenti schede DD per definire delle librerie:

```
//SYSLIB   DD  DSN=SYS1.FORTLIB,DISP=SHR
//         DD  DSN=SYS1.FORT2,DISP=SHR
//         DD  DSN=SYS1.SSPLIB,DISP=SHR
```

Si vuole:
a) Forzare la ricerca della libreria SYS1.FORT2 su un disco privato MYDISK.
b) Aggiungere una nuova libreria MYLIB.

E' necessario allora codificare le seguenti schede:

```
//LKED.SYSLIB  DD
//             DD  UNIT=SYSDA,VOL=SER=MYDISK,DISP=CID
//             DD
//             DD  DSN=MYLIB,DISP=OLD,UNIT=SYSDA,VOL=SER=MYDISK
```

La prima scheda DD, non fornendo alcun parametro, lascia validi quelli preesistenti sulla vecchia prima scheda.
La seconda scheda DD cambia il valore DISP=SHR in DISP=OLD, aggiunge i parametri UNIT e VOL, e lascia invariato il parametro DSN=SYS1.FORT2.
La terza scheda lascia invariati tutti i parametri preesistenti.
La quarta scheda si aggiunge alle tre preesistenti, definendo la nuova libreria MYLIB.

- PROCEDURE CATALOGATE FORTRAN SCITO CS

E' stato unificato, nelle procedure catalogate Fortran G1 ed H-Esteso (FORTG e FORTX) che contengono i passi di link edit o loader, l'ordine delle librerie definite dalla scheda SYSLIB.
La scheda SYSLIB e' ora cosi' formata:

```
//SYSLIB   DD  DDNAME=FIRSTLIB
//         DD  DSN=SYS1.FXLIB,DISP=SHR
//         DD  DSN=SYS1.FORT2,DISP=SHR
//         DD  DSN=SYS1.PICTLIB,DISP=SHR
//         DD  DSN=SYS1.SSPLIB,DISP=SHR
//         DD  DSN=SYS1.CERNLIB,DISP=SHR
//         DD  DDNAME=LASTLIB
//FIRSTLIB DD DSN=SYS1.NULLPDS,DISP=SHR
//LASTLIB  DD DSN=SYS1.NULLPDS,DISP=SHR
```

Le schede FIRSTLIB e LASTLIB consentono rispettivamente l'inserimento in testa e in coda alle altre, di una propria libreria di programmi, senza la necessita' di ricodificare tutte le altre schede DD: la sola esigenza e' che tale libreria abbia, se inserita in testa, lo stesso formato e bloccaggio di quelle che seguono. Un bloccaggio inferiore causa infatti l'allocazione, da parte del sistema, di un buffer troppo piccolo per contenere i records fisici delle altre librerie; in tal caso la libreria privata deve essere necessariamente codificata in coda alle altre.

Si ricorda inoltre che la ricerca delle subroutines richiamate dal programma, avviene sequenzialmente nelle varie librerie, per cui si deve fare attenzione a subroutines che hanno lo stesso nome, ma che compaiono in librerie diverse: cosi' ad esempio se una propria libreria MYLIB contiene la subroutine SYMBOL, che compare anche nella libreria PLOTLIB si possono avere vari casi:

1) Il programma vuole richiamare la subroutine SYMBOL contenuta nella libreria MYLIB e puo' fare uso di tutte le altre subroutines contenute nelle altre librerie:

```
// EXEC FORTG
//COMP.SYSIN DD *
.... programma ....
/*
//GO.FIRSTLIB DD DSN=MYLIB,DISP=CLD,...
```

Si noti che in questo caso le due codifiche:

```
//GO.FIRSTLIB DD ....        e      //GO.SYSLIB DD ....
```

sarebbero equivalenti.

2) Il programma vuole richiamare la subroutine SYMBOL contenuta nella libreria PLOTLIB, invece di quella contenuta in MYLIB, e puo' fare uso di tutte le altre subroutines contenute nelle altre librerie:

```
// EXEC FORTG
//COMP.SYSIN DD *
.... programma ....
/*
//GO.LASTLIB DD DSN=MYLIB,DISP=CLD,...
```

3) Il programma vuole richiamare la subroutine SYMBOL contenuta nella libreria MYLIB e vuole nello stesso tempo eliminare qualsiasi riferimento alle subroutines contenute nella libreria PLOTLIB:

```
// EXEC FORTG
//COMP.SYSIN DD *
.... programma ....
/*
//GO.SYSLIB  DD
//           DD
//           DD
//           DD  DSN=MYLIB,DISP=CLD,...
```

Nota: cambiamenti simili sono avvenuti anche nelle procedure del PL/1, del CSMP, del Plotter, del Linkage Editor e del Loader.
Se si fa uso particolare di tali procedure (per quanto riguarda la scheda SYSLIB) si consiglia di prendere visione delle variazioni presso la Sezione Consulenza del Centro.

- JCL e unita' nastro

Il sottoparametro DEN del parametro DCB serve a selezionare l'opportuna densita' di lettura o scrittura:

l'omissione del sottoparametro DEN, causa la scelta, in fase di scrittura, da parte del sistema, della densita' piu' alta esistente sull'unita' richiesta, e cioe':

UNIT=TPE9	(senza DEN)	implica	1600 Bpi
UNIT=TPV9	(senza DEN)	implica	1600 Bpi
UNIT=TPH9	(senza DEN)	implica	6250 Bpi

Si consigliano vivamente gli Utenti di codificare sempre esplicitamente il sottoparametro DEN del parametro DCB senza affidarsi alla densita' default.
Si evitera' in tal modo qualsiasi difformita' nella scrittura e lettura dello stesso nastro dovuta ad una richiesta di montaggio su unita' diverse.
Si eviteranno inoltre errori nel caso che, in futuro, per esigenze di sistema, dovessero eventualmente cambiare le associazioni dei nomi simbolici (TPE9, TPV9) con le unita' sopra descritte.

Esempio:

Richiesta di		Codifica consigliata
9 piste	800 Bpi	UNIT=TPE9,DCB=DEN=2
	1600 Bpi	UNIT=TPV9,DCB=DEN=3
	6250 Bpi	UNIT=TPH9,DCB=DEN=4

- Opzione DEFER

La richiesta di nastri in OS avviene tramite la codifica di schede SETUP, MESSAGE e DD.
Il jcb viene eseguito al momento in cui tali risorse sono disponibili e le risorse richieste vengono allocate alla partenza di ogni singolo step del programma.
In particolare ad esempio, la richiesta all'operatore di

montaggio di un nastro, avviene quando inizia lo step di GO del programma che richiede tale nastro.
Puo' pero' accadere che il programma in oggetto sia strutturato in modo da compiere un certo numero di elaborazioni diverse, prima di eseguire le effettive istruzioni riguardanti il nastro (OPEN, READ, WRITE ecc.): per tutto tale periodo il nastro resta allocato ma inutilizzato.
Inoltre puo' accadere che il programma considerato vada in ABEND prima delle istruzioni relative all'utilizzo del nastro, rendendo cosi' inutile la ricerca dello stesso e il montaggio da parte dell'operatore addetto.
Esiste pertanto in OS la possibilita' di richiedere il montaggio di un nastro solo al momento in cui viene eseguita la prima istruzione che lo interessa: e' per questo prevista la codifica del sottoparametro DEFER nel parametro UNIT della scheda DD in oggetto. Esempio:
UNIT = (TPE9,,DEFER)
L'uso di tale parametro, oltre ai motivi suddetti e' anche consigliabile in quanto conserte una maggiore sicurezza nell'uso dei nastri: evita infatti il montaggio del nastro dopo che il job che lo ha richiesto sia gia' terminato per ABEND, lasciando cosi' il nastro disponibile per un qualsiasi altro job, col grave rischio che il nastro venga distrutto.

Allocazioni librerie private su disco.

- Sotto OS esiste un insieme di programmi per la gestione automatica di data sets su disco: tali programmi controllano periodicamente la situazione di tutti i dischi presenti sul sistema e prendono opportune decisioni per quanto concerne la liceita' o meno delle allocazioni, il loro addebito ecc..ecc..
  Pertanto, in teoria, l'uso di tali programmi di controllo lascerebbe libero l'Utente di creare o distruggere a piacimento i propri data sets sui dischi autorizzati, senza curarsi di dover ogni volta avvertire il Centro delle variazioni apportate.

- Onde evitare pero' allocazioni improprie o con codici non permessi, e' stato ritenuto opportuno far restare presso il Centro una documentazione relativa alla richiesta da parte di un certo Ente di una data set con un certo nome. Pertanto le richieste di autorizzazione alla allocazione di data sets devono sempre essere effettuate, sugli appositi moduli, presso lo Sportello Utenti.
  Una volta ottenuta l'autorizzazione, l'Utente puo' chiedere che il suo data set gli venga anche fisicamente allocato sul disco (e in tal caso dovra' fornire tutte le indicazioni riguardanti lo spazio richiesto e il formato di tale data set) oppure puo' allocarlo da solo e variarne a piacimento le caratteristiche.
  L'addebito del data set, partira' solo da quando questo sara', effettivamente allocato su disco e non dal giorno in cui viene richiesta l'autorizzazione. Analogamente se l'Utente decide di eliminare da disco il suo data set per tre giorni e poi ricrearlo sempre con lo stesso nome, ma con caratteristiche diverse non ha bisogno di informare lo Sportello Utenti: i programmi di controllo provvedono a far si' che il tempo per cui il data set non esiste fisicamente su disco, non venga addebitato.

- Vi sono due tipi di modalita' per la richiesta di autorizzazione all'allocazione su disco:

  a) Autorizzazione del codice di lavoro XXXX.
     Un Ente puo' chiedere, tramite un suo rappresentante, che un suo codice venga abilitato a tutte le allocazioni sui dischi permessi.
     Cio' significa che i programmatori di tale Ente sono autorizzati ad allocare sui dischi permessi, qualsiasi data set sia della forma: XXXXYYZZ (nome in forma standard), dove:
     XXXX e' il codice di lavoro dell'Ente
     YYZZ sono quattro caratteri alfanumerici a piacimento di cui in genere i primi due rappresentano il codice del programmatore e gli altri due identificano i vari data sets di uno stesso programmatore.

Questo permette una completa flessibilita' nelle allocazioni e disallocazioni di data sets, lasciando i programmatori dell'Ente completamente liberi di creare o distruggere data sets con nome standars; tutti i data sets di nome XXXXYYZZ trovati sui dischi permessi, verranno addebitati automaticamente all'Ente sotto il codice di lavoro XXXX per la durata della loro esistenza su disco.

Quando si voglia togliere l'autorizzazione del codice di lavoro XXXX, bisogna darne comunicazione allo Sportello Utenti. In questo caso saranno considerati non autorizzati tutti i data sets di nome XXXXYYZZ allocati in precedenza.

b) Autorizzazione del nome del data set.
Quando l'Utente ha necessita' di allocare data sets con nome non standard o preferisce per motivi di protezione non richiedere l'autorizzazione del codice di lavoro, puo' effettuare una richiesta di autorizzazione per un particolare data sets fornendo allo Sportello Utenti:
- il codice di lavoro dell'ente di appartenenza (al quale verra' addebitato il data set)
- il nome del data set (che potra' a piacimento essere standard o non standard)
- il disco su cui vuol far risiedere tale data set (USERxx -3350- per data sets che occuperanno meno di 5 cilindri, MVSxxx -3380- per data sets che occuperanno piu' di 5 cilindri).

Inoltre puo' richiedere che tale data set gli venga fisicamente allocato su disco dal personale del Centro (e in tal caso dovra' fornire gli altri parametri necessari: SPACE e DCB) oppure puo' allocarlo personalmente con i parametri che preferisce, tenendo conto dell'unica limitazione del numero di cilindri.

Essi potranno inoltre in qualsiasi momento variarne le caratteristiche eliminandolo e ricreandolo, senza per questo dover ripassare attraverso lo Sportello Utenti, purche' lascino invariato il nome del data set.
Si noti che dal momento della autorizzazione al momento della creazione del data set (o dal momento di una sua cancellazione fino al momento della sua ri-creazione) non deve intercorrere piu' di un mese. In caso contrario l'autorizzazione viene automaticamente eliminata e va richiesta nuovamente allo Sportello Utenti.

Quando tale data set non sara' piu' loro necessario, ne daranno comunicazione allo Sportello Utenti, chiedendo di toglierne l'autorizzazione. Anche in questo caso l'eliminazione fisica del data set da disco puo' essere effettuata personalmente da loro o essere delegata al personale dello Sportello.

- Le modalita' di allocazione, disallocazione fisica di un data set su disco sono le seguenti:

  a) un data set viene fisicamente allocato su disco tramite le schede:

```
// EXEC PGM=IEFBR14
//nome DD DSN=xxxxyyzz,DISP=(NEW,KEEP),
//       UNIT=SYSDA,VOL=SER=volume,
//       SPACE=(spazio),DCB=(formato)
```

  e viene richiamato con una scheda del tipo

```
//nome DD DSN=xxxxyyzz,DISP=(OLD,KEEP),
//       UNIT=SYSDA,VOL=SER=volume
```

  b) un data set viene fisicamente eliminato da un disco tramite schede del tipo:

```
//        EXEC     PGM=IEFBR14
//nome      DD     DSN=xxxxyyzz,DISP=(OLD,DELETE),
//        UNIT=SYSDA,VOL=SER=volume
```

- Per poter controllare i propri data sets allocati su disco, si hanno a disposizione vari programmi, di cui diamo brevemente le specifiche:

  a) Lista dei data sets specificati e allocati su un dato disco (nel caso di data sets partitioned vengono anche fornite le informazioni sul directory e sui membri del data set):

```
//    EXEC    LISTA,V=volume,C='*',D=nome
```

  b) Lista di tutti i data sets autorizzati al proprio codice di lavoro, ma non necessariamente allocati:

```
//    EXEC    LISTA,C=DSMIEI
```

  Il programma fornisce, come risposta, la situazione di tutti i data sets autorizzati ed addebitati al codice di lavoro che compare sulla scheda JOB.
  Non verranno listati da tale programma i data sets non autorizzati.
  La stampa comprende le seguenti informazioni:

DSNAME	(44 caratteri):	nome del data set autorizzato. Se finisce con asterisco significa che sono autorizzati tutti i data sets che cominciano per tale sequenza di caratteri.
VOLUME	(6 caratteri):	nome del volume su cui risiede il data set.
NASCITA	(9 caratteri):	data di nascita del data set se e' stato allocato o data di avvenuta autorizzazione.
MORTE	(9 caratteri):	eventuale data di cancellazione del data set.

ULTIMO_ADD (9 caratteri): data di ultimo addebito (2) eseguito sul data set, se e' stato addebitato, altrimenti nulla o una data uguale a quella di nascita.

Si ricorda che un data set viene addebitato tutte le volte che:
a) viene cancellato
b) subisce una variazione di spazio in piu' o in meno
c) sono trascorsi piu' di 15 giorni dall'ultimo addebito.

Note: Possono comparire sull'unita', varie altre informazioni supplementari come ad esempio: DA NASCERE significa che il data set e' autorizzato, ma non allocato; VIVO significa che il data set e' autorizzato ed allocato; MORTO significa che il data set e' stato autorizzato, allocato e cancellato; AUTORIZZAZIONE PERMANENTE significa che l'autorizzazione ad allocare data sets del nome indicato rimane attiva anche quando il data set sara' cancellato e comunque fino a quando non viene fatta un'apposita richiesta di eliminarla.

Vengono inoltre date le informazioni relative a quali volumi possono contenere data sets di nome standard, cioe' quelli il cui nome inizia con il codice di addebito.

Dal momento in cui il data set e' stato autorizzato, puo' essere fisicamente allocato su disco; un data sets allocato ma non preventivamente autorizzato viene automaticamente cancellato.
E' pertanto estremamente importante che ciascun Utente che voglia creare delle proprie librerie non temporanee su disco, chieda la preventiva autorizzazione e si accerti che questa sia stata data, prima di allocare fisicamente il data set sul disco interessato.

- Dischi di lavoro utilizzabili senza alcuna formalita' da espletare:

- Considerazioni sui fattori di bloccaggio

Il parco dischi installato al CNUCE comprende unita' di tipo 3350 e unita' di tipo 3380; l'unita Mass Storage utilizza poi supporti virtuali di tipo 3330 modello 1.

Diamo qui sotto per completezza, un raffronto tra le caratteristiche tecniche dei vari modelli:

	3330/1	3350	3380
MB/pack	100	317.5	630.2
cyl/pack (+alternati)	404+7	555+5	885+1
trk/cyl	19	30	15
bytes/trk	13030	19069	47476
vel. trasf. (Kb/sec)	806	1198	3000
seek medio (ms)	30	25	16
search medio (ms)	8.4	8.4	8.3

Per quello che riguarda problemi di programmazione, ha particolare interesse il numero di bytes per traccia (13030 per i 3330, 19069 per i 3350 e 47476 per i 3380).

Infatti in base al fattore di bloccaggio scelto per i propri data sets, si puo' verificare uno spreco piu' o meno maggiore di bytes all'interno di una traccia, dovuto alla rimanenza alla fine della traccia, di spazio non utilizzato in quanto non sufficiente a contenere un ulteriore blocco di dati.

Nel fare i conti di tale spreco, si devono tenere presenti due quantita':

1) Capacita' massima in bytes della traccia.
Una traccia e' costituita da informazioni riservate al sistema (Index Point, Home Address, Inter Block Gap, Inter Record Gap ecc.) piu' un insieme di bytes effettivamente disponibili per i dati. L'insieme di tutti i campi fornisce la capacita' totale T della traccia, che e':

	3330	3350	3380
capacita' totale T (bytes)	13165	19254	47968

2) Numero di blocchi per traccia.

Ciascun blocco di dati che deve essere memorizzato su disco e' a sua volta composto da una parte di informazioni fisse piu' la lunghezza dei dati effettivi, piu' la lunghezza dell'eventuale chiave per data sets con indici:

		3330	3350	3380
informazioni fisse CL (bytes)	records senza chiave	135	185	480
	records con chiave	191	267	704

Pertanto indicando con DL la lunghezza dei dati, KL la lunghezza della chiave e CL la lunghezza delle informazioni costanti, si ha per il numero N di blocchi per traccia (T = capacita' totale della traccia) tenendo conto anche delle Inter Block Gap e di tutte le altre informazioni riservate al sistema la seguente formula:

$$N = \frac{T}{CL + DL + KL}$$

Nella formula si deve tener conto che per i 3380, al contrario degli altri modelli, i valori di DL e KL vanno incrementati di 12 e arrotondati verso l'alto al piu' vicino multiplo di 32.

Servendoci delle due formule precedenti, si puo' essere in grado di determinare, per ogni organizzazione (PS, IS ecc.) e per ogni formato (FB, U ecc.) di data sets, il residuo dei bytes all'interno della traccia.

Senza entrare nei dettagli del calcolo, si riportano qui i risultati validi per due formati tipici:

1) DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=.....

Nel caso di data sets fissi-bloccati con lunghezza del record logico di 80 bytes i valori del BLKSIZE che portano a bassi sprechi (inferiori ai 100 bytes per traccia in assoluto e con una occupazione dei dati significativi almeno del 90%) sono rispettivamente:

3330	3350	3380
2480, 4240, 6400, 12960	2560, 9440, 19040	3840, 23440, 47360, 47440

Tenendo inoltre conto che:

a) un basso fattore di bloccaggio implica maggiori operazioni di I/O;

b) alti fattori di bloccaggio necessitano di buffer di notevoli dimensioni di memoria;

c) il sistema OS non ammette l'uso di bloccaggi superiori a 32768;

d) la gestione della memoria virtuale a "pagine" consiglia di mantenersi sempre nell'ambito di valori multipli di 4K (leggermente inferiori per permettere l'inserimento nella stessa pagina, oltre al buffer, anche di alcuni blocchi di controllo di sistema), ne consegue che i valori consigliati per tali tipi di data sets sono:

3330	3350	3380
BLKSIZE = 2480	BLKSIZE = 2560	BLKSIZE = 3860

Da sconsigliare fermamente invece per tutte le unita' il bloccaggio 3200 (spesso usato) in quanto porta a sprechi decisamente superiori.

L'eventuale spostamento di data sets di tale tipo da un modello di disco all'altro dovra' naturalmente comportare, per una massima ottimizzazione, un cambiamento del bloccaggio nel modo descritto.
Se per qualsiasi motivo si fosse costretti a mantenere sui vari dischi dei data sets con lo stesso bloccaggio, tra i valori elencati in precedenza, l'unico che mantiene basso lo spreco su tutti i dischi e' quello di 2480.

2) DSORG=PO,RECFM=U,BLKSIZE=.....

Nel caso di data sets fissi o undefined, ma tutti della stessa lunghezza, si ha per i valori del bloccaggio sui due dischi:

Blocchi per trk	3330 DA (bytes) A	3350 DA (bytes) A	3380 DA (bytes) A
15	688 --> 742	1019 --> 1098	2485 --> 2676
14	743 --> 805	1099 --> 1190	2677 --> 2932
13	806 --> 877	1191 --> 1296	2933 --> 3188
12	878 --> 962	1297 --> 1419	3189 --> 3476
11	963 --> 1061	1420 --> 1565	3477 --> 3860
10	1062 --> 1181	1566 --> 1740	3861 --> 4276
9	1182 --> 1327	1741 --> 1954	4277 --> 4820
8	1328 --> 1510	1955 --> 2221	4821 --> 5492
7	1511 --> 1745	2222 --> 2565	5493 --> 6356
6	1746 --> 2059	2266 --> 3024	6357 --> 7476
5	2060 --> 2498	3025 --> 3665	7477 --> 9076
4	2499 --> 3156	3666 --> 4628	9077 --> 11476
3	3157 --> 4253	4629 --> 6233	11477 --> 15476
2	4254 --> 6447	6234 --> 9442	15477 --> 23476
1	6448 --> 13030	9443 --> 19069	23477 --> 47476

Le considerazioni da fare per tali data sets sono simili a quelle fatte precedentemente, con l'aggiunta che ora il bloccaggio ottimale e' quello "alla traccia" (ricordiamo pero' la limitazione dell'OS ad un massimo di 32768), cioe':

3380	3380	3380
BLKSIZE = 13030	BLKSIZE = 19069	BLKSIZE = 47476

Considerando pero' anche il fattore di paginazione discusso precedentemente, si puo' consigliare anche un bloccaggio inferiore:

3330	3350	3380
BLKSIZE = 3156	BLKSIZE = 3665	BLKSIZE = 3960

Analogamente a quanto detto in precedenza, se si dovesse usare un bloccaggio unico inferiore ai 4K per tutti i dischi, sarebbero da preferire i maniera pressoche' equivalente i valori 3156 e 3665.

2f) Schede particolari

- Scheda /*

  Indica la fine di un data set che inizia con una scheda DD * o DD DATA senza la codifica esplicita di un particolare delimitatore.

- Scheda //*

  E' una scheda commento: qualsiasi informazione codificata da colonna 4 a colonna 80 non viene elaborata dal sistema operativo, ma viene semplicemente listata in uscita tra le altre schede controllo.

3 - CONSIDERAZIONI PARTICOLARI

Questo capitolo si riferisce interamente al vecchio sistema OS/SVS e non ha piu' validita' nel sistema MVS; viene pero' ugualmente lasciato per completezza e soprattutto perche' le informazioni in esso contenute contribuiscono ad approfondire e chiarire il funzionamento di alcuni parametri della scheda DD.

3a) Data sets dedicati

Ricordiamo come tutti i programmi di utilita' del sistema operativo OS (Compilatori, Assemblatori, Linkage Editor, Loader, ecc.) facciamo uso per le loro funzioni, di data sets di comodo che vengono via via allocati su aree di lavoro temporanee su disco.
Cio' comporta per ciascun job, un certo tempo speso nella allocazione, apertura, chiusura e disallocazione di tali data sets.
Si e' pertanto pensato di utilizzare una tecnica che permette il risparmio di tutto il tempo sopra descritto e che si basa appunto sull'uso di "data sets dedicati".
Ricordiamo che con il sistema operativo OS/HASP sono presenti in memoria un certo numero di initiators, che vengono fatti partire al momento dell'IPL del sistema operativo e rimangono attivi fino al successivo IPL.
Ciascuno di tali initiators elabora, nel corso della sessione di lavoro un certo numero di jobs, uno dietro l'altro.
Cio' che il sistema operativo OS permette e' di far si' che ciascun initiator, al momento della partenza, allochi quelle aree di lavoro che successivamente saranno utilizzate da tutti i job elaborati da quel dato initiator.
Tali aree saranno cosi' disponibili per tutti i jobs fino alla fine della sessione, senza che siano necessarie continue allocazioni e disallocazioni da parte dei jobs stessi.
E' inoltre possibile dedicare anche data sets non temporanei, quali ad esempio le librerie di sistema: si evitera' in tal caso ogni volta la ricerca di tali librerie tramite il catalogo.

Procedura INIT

```
//IEFPROC  EXEC  PGM=IEFIIC,PARM='A,LIMIT=13'
//SYSUT1 DD DSN=&&UT1,DISP=(NEW,DELETE),
// UNIT=SYSDA,VOL=SER=VSLIB1,SPACE=(CYL,(7))
//SYSUT2 DD DSN=&&UT2,DISP=(NEW,DELETE),
// UNIT=SYSDA,VOL=SER=LIBMVT,SPACE=(CYL,(7))
//SYSUT3 DD DSN=&&UT3,DISP=(NEW,DELETE),
// UNIT=SYSDA,VOL=SER=LIBMVT,SPACE=(CYL,(7))
//LOADSET DD DSN=&&LOAD,DISP=(NEW,DELETE),
// UNIT=SYSDA,VOL=SER=LIBMVT,SPACE=(CYL,(5))
```

Si nota, nella procedura di initiator sopra riportata che:

a) sono dedicati i data sets &UT1, &UT2, &UT3, &LOAD
b) il parametro DISP per data sets temporanei e' (NEW,DELETE), che comporta la loro creazione al momento della partenza dell'initiator e la loro distruzione solo quando l'initiator viene fermato, mentre per quelli permanenti e' (SHR,KEEP)
c) non e' codificato il parametro DCB

Procedura FORTG

```
//FORTGCG PROC DECK=NODECK,LIST=NOLIST,MAP=MAP,SOURCE=SOURCE,CODE=BCD,
//            LET=NOLET,CALL=CALL,CONDC=4,RGO=128K,RCOMP=128K,
//            PROG=IGIFORT
//COMP   EXEC  PGM=&PROG,PARM='&DECK,&LIST,&MAP,&SOURCE,&CODE',
//            REGION=&RCOMP
//STEPLIB DD DSN=SYS1.FGLINK,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSPUNCH DD  SYSOUT=B
//SYSUT1 DD DSN=&SYSUT1,DISP=(NEW,PASS),
//   UNIT=SYSDD,SPACE=(13030,(133))
//SYSUT2 DD DSN=&SYSUT2,DISP=(NEW,PASS),
//   UNIT=SYSDD,SPACE=(13030,(133))
//SYSLIN  DD DSN=&LOADSET,DISP=(NEW,PASS),SPACE=(13030,(95)),
//  UNIT=SYSDD,DCB=(RECFM=FB,LRECL=80,BLKSIZE=2480,BUFNO=2)
//GO     EXEC  PGM=LOADER,PARM=(&LET,&MAP,&CALL,'SIZE=750K'),
//            COND=(&CONDC,LT,COMP),REGION=&RGO
//SYSLIB   DD  DSNAME=SYS1.FXLIB,DISP=SHR
//       DD    DSN=SYS1.SSPLIB,DISP=SHR
//         DD  DSNAME=SYS1.FORT2,DISP=SHR
//SYSLIN  DD DSN=&LOADSET,DISP=(OLD,PASS),UNIT=SYSDD
//         DD  DDNAME=OBJECT
//SYSLOUT  DD  SYSOUT=A
//FT01F001 DD DSN=&SYSUT1,DISP=(OLD,PASS),UNIT=SYSDD,
//   DCB=(RECFM=VBS,BLKSIZE=256)
//FT02F001 DD DSN=&SYSUT2,DISP=(OLD,PASS),UNIT=SYSDD,
//   DCB=(RECFM=VBS,BLKSIZE=256)
//FT03F001 DD DSN=&SYSUT3,DISP=(NEW,PASS),
//   UNIT=SYSDD,SPACE=(13030,(133)),
//   DCB=(RECFM=VBS,BLKSIZE=256)
//FT04F001 DD DSN=&LOADSET,DISP=(NEW,PASS),
//   UNIT=SYSDD,SPACE=(13030,(95)),
//   DCB=(RECFM=VBS,BLKSIZE=256)
//FT05F001 DD  DDNAME=SYSIN
//FT06F001 DD  SYSOUT=A
//FT07F001 DD  SYSOUT=B
```

La procedura indicata nella pagina precedente e' codificata in modo tale da far uso dei data sets dedicati definiti in precedenza.
Si notino i seguenti punti:

a) il parametro DSNAME non contiene il nome del data set temporaneo, ma il nome della DD (preceduto dal carattere &) che definisce il data set dedicato che si vuol utilizzare

b) la definizione dello spazio e' fatta in blocchi, anziche' in TRK o CYL

c) e' esplicitamente indicato il parametro DCB

d) particolari considerazioni saranno fatte in seguito per il parametro DISP

L'uso dei data sets dedicati allocati dall'initiator sara' costituito da una normale allocazione di data sets temporanei da parte del job in esecuzione se si verifica una delle seguenti condizioni:

a) il parametro DSNAME non indica la scheda DD della procedura di initiator

b) lo spazio richiesto e' definito in TRK o CYL

c) lo spazio totale (primario e secondario) e' richiesto in blocchi, ma supera lo spazio totale (primario e secondario) definito nella procedura di initiator

d) il numero di blocchi di directory richiesti eccede il numero di quelli definiti nella procedura di initiator, o e' presente mentre prima non lo era o viceversa

Si noti inoltre che:

a) se e' codificata una quantita' secondaria nel parametro SPACE, deve obbligatoriamente essere codificato il parametro BLKSIZE in quanto tale valore viene utilizzato dal sistema per calcolare il valore dell'allocazione secondaria

b) mentre l'allocazione del data set viene fatta tenendo conto della quantita' primaria definita nella procedura di initiator, la quantita' secondaria eventualmente specificata nella procedura di uso avra' la prevalenza nel momento in cui si presenta la necessita' di estendere tale spazio

c) gli incrementi di spazio eventualmente allocati da un job ai data sets dedicati, rimarranno tali fino a che non verra' fermato il relativo initiator

d) quando un data set partitioned dedicato raggiunge una condizione di fine volume l'initiator relativo va in ABEND.

Considerazioni sul parametro DISP nelle procedura di uso dei data sets dedicati.

Come si e' gia' detto, i data sets dedicati rimangono attivi per tutta la durata di una sessione, fino a che non si ferma l'initiator ad essi relativo.
Per questo la codifica del parametro DISP nella procedura catalogata di uso viene automaticamente variata dal sistema operativo, a seconda della seguente tabella:

NEW	-->	OLD	,KEEP	-->	,KEEP
OLD/SHR	-->	OLD	,PASS	-->	,PASS
MOD	-->	MOD	,DELETE	-->	,KEEP

Studiamo ora dei casi particolari in cui tali variazioni possono far subire delle modifiche nell'esecuzione di un programma rispetto all'utilizzazione di data sets non dedicati.

Considerazioni sul parametro DISP nei data sets &&LOADSET (sequenziale) e &&GOSET(GO) (partitioned) rispettivamente nelle schede DD SYSLIN e SYSLMOD in relazione ai data sets dedicati.

<u>COMPILAZIONE</u>

```
//SYSLIN DD DSN=&&LOADSET,DISP=...
```

Uso dei data sets non dedicati:

L'uso di data sets non dedicati impone, in fase di compilazione, la creazione del data set &&LOADSET e quindi una DISP=(NEW,PASS) o DISP=(MOD,PASS) nella procedura catalogata relativa.
In particolare dobbiamo ricordare che spesso capita di dover eseguire due o piu' compilazioni successive (esempio quando si abbia un programma principale in Fortran e delle subroutines in Assembler) seguite da un unico passo di Linkage Editor ed Esecuzione. In tal caso, mentre il primo passo di compilazione necessita di una DISP=(NEW,PASS) o DISP=(MOD,PASS), il secondo e i seguenti passi di compilazione dovranno avere una

DISP=(MOD,PASS), in modo che i moduli oggetto dei sottoprogrammi si aggiungano a quello del programma principale e non si sovrappongano ad esso.
Pertanto, mentre procedure catalogate che avessero DISP=(MOD,PASS), potrebbero essere usate senza alcuna modifica, per l'uso di procedure catalogate con DISP=(NEW,PASS) si dovrebbe procedere ad un override della scheda corrispondente con i parametri:
//COMP.SYSLIN DD DISP=(MOD,PASS)

Uso dei data sets dedicati:
L'uso dei data sets dedicati comporta l'utilizzo di data sets esistenti gia' su disco, e quindi l'impossibilita' di usare DISP=(MOD,PASS), in quanto tali data sets potrebbero gia' contenere, nella prima compilazione, informazioni non volute.
E' pertanto necessario avere codificato, nelle procedure catalogate DISP=(NEW,PASS), che equivale, in questo caso, a una specifica DISP=(OLD,KEEP).
Nel caso di piu' compilazioni consecutive seguite da un unico passo di Linkage Editor ed Esecuzione, occorre pertanto sempre codificare, nel secondo e nei passi di compilazione seguenti la scheda:
//COMP.SYSLIN DD DISP=(MOD,PASS)

LINKAGE EDITOR
//SYSLMOD DD DSN=&&GOSET(&NAME),DISP=...

Per quanto concerne le specifiche da attribuire al data set &&LOADSET che contiene i moduli oggetto da elaborare, si deve semplicemente rilevare che, alla fine della fase di Linkage Editor, tale data set diviene inessenziale, e quindi sara' codificato nelle procedure catalogate il parametro DISP=(OLD,DELETE), che, nel caso di data sets dedicati, equivale a (OLD,KEEP), in quanto il data set non viene fisicamente distrutto, ma continua ad esistere anche dopo la fine del job per essere utilizzato in altre elaborazioni.

Uso dei data sets non dedicati:
Valgono le stesse considerazioni fatte in precedenza per il data set &&LOADSET, e quindi una DISP=(NEW,PASS) o DISP=(MOD,PASS).
Capita pero' talvolta di dover effettuare, nell'ambito di uno stesso job, due fasi di Compile Link e Go consecutive. In tal caso si devono distinguere le due situazioni.
La procedura catalogata presenta il parametro DISP=(NEW,PASS): ne consegue che anche la seconda fase di Linkage Editor fa riferimento a tale parametro, causando errore, perche' il data set definito come NEW

e' in realta' gia' stato creato nella prima fase di Linkage Editor e quindi esiste gia' su disco (Soluzione: aggiungere in fase di esecuzione, una scheda del tipo
//GO.SYSLMOD DD DSN=&&GOSET, DISP=(OLD,DELETE)).
La procedura presenta il parametro DISP=(MOD,PASS): nella prima fase di Linkage Editor, il data set non esiste ancora, MOD viene assunto equivalente a NEW, e il data set viene regolarmente creato. Nella seconda fase poi MOD, riferendosi a un data set partitioned eguivale a OLD, viene ritrovato il data set precedente, ma si ha di nuovo una condizione di errore, in quanto, all'interno di tale data set, cerca di creare un membro, GO, con lo stesso nome del precedente. (Soluzioni: aggiungere, in fase di esecuzione, una scheda del tipo
//GO.SYSLMOD DD DSN=&&GOSET,DISP=(OLD,DELETE)
oppure richiamare le due procedure di Compile Link e Go, formendo alla prima il parametro NAME=GO1 e alla seconda il parametro NAME=GO2, in modo che vengano creati, nelle due fasi di Linkage Editor, due membri diversi (GO1 e GO2) all'interno del data set &&GOSET.

Uso dei data sets dedicati:
Valgono le considerazioni fatte in precedenza per il data set &&LOADSET, e quindi la necessita' di codificare solo un parametro DISP=(NEW,PASS), che equivale, in questo caso, a una specifica DISP=(OLD,KEEP). Nell'esecuzione di due fasi consecutive di Compile Link e Go si ha lo stesso errore che nel caso di data sets non dedicati e cioe' il tentativo di inserire, nel data set partitioned &&GOSET, due membri con lo stesso nome GO.
Pero', al contrario di prima, la soluzione ora e' una sola: non e' infatti possibile aggiungere in fase di esecuzione la scheda
//GO.SYSLMOD DD DSN=&&GOSET, DISP=(OLD,DELETE)
in quanto, essendo i data sets dedicati, esistono sempre su disco, e tale specifica verrebbe automaticamente cambiata dal sistema in DISP=(OLD,KEEP). Unica soluzione resta pertanto quella di codificare, nelle due fasi di Linkage Editor, i parametri NAME=GO1 e NAME=GO2.

Diamo qui di seguito una tabella riassuntiva di quanto fin qui chiarito, con le specifiche da adottare nei due casi di compilazioni multiple (C-C-LG) e di Compile Link e Go consecutivi (CLG-CLG) tenendo conto che, nel nostro Centro, il data set &&LOADSET e' dedicato, mentre &&GOSET non lo e':

	FASE	DD	DSN	DISP	C-C-LG	CLG-CLG
Data sets non dedicati	COMP	SYSLIN	&&LOADSET	(NEW,PASS)	NEW nella prima, MOD nelle seguenti	OK
				(MOD,PASS)	OK	OK
		SYSLIN	&&LOADSET	(OLD,DELETE)	OK	OK
	LKED			(NEW,PASS)	OK	Aggiungere DELETE in fase di GO
		SYSLMOD	&&GOSET			
				(MOD,PASS)	OK	Aggiungere DELETE in fase di GO oppure NAME=GO1,NAME=GO2
Data sets dedicati	COMP	SYSLIN	&&LOADSET	(NEW,PASS)	NEW nella prima, MOD nelle seguenti	OK
				(MOD,PASS)	NO	NO
		SYSLIN	&&LOADSET	(OLD,DELETE)	OK	OK
	LKED			(NEW,PASS)	OK	NAME=GO1,NAME=GO2
		SYSLMOD	&&GOSET			
				(MOD,PASS)	NO	NO

3b) Miscellanea

Come abbiamo gia' detto le procedure catalogate contengono le schede DD che definiscono i data sets usati dal programma richiamato.
Puo' a volte essere utile da parte dell'Utente usare le procedure catalogate del Centro, ma aggiungere delle proprie schede DD o sostituire delle schede DD esistenti con altre proprie: in tal caso il ddname deve essere qualificato, facendolo precedere dal nome della scheda EXEC che definisce quel passo (COMP, LKED, GO).
Facciamo degli esempi che chiariscano quanto detto:

- Compilazione ed esecuzione di un programma Assembler che legge delle schede dati e stampa su printer. Si noti la relazione che intercorre tra il ddname ed il nome del file trattato nel programma:

```
//ASMJOB   JOB  ...informazioni di accounting...
//        EXEC  ASMF
//COMP.SYSIN DD *
MAIN      START
          ....
          OPEN  (DCBIN,,DCBOUT,(OUTPUT))
          ....
          GET   DCBIN,WORKA
          PUT   DCBOUT,WORKA
          ....
          CLOSE (DCBIN,,DCBOUT)
          ....
          RETURN
WORKA     DS    CL80
DCBIN     DCB   DSORG=PS,MACRF=(GM),DDNAME=DDIN
DCBOUT    DCB   DSORG=PS,MACRF=(PM),DDNAME=DDOUT
          END   MAIN
/*
//GO.DDOUT DD  SYSOUT=A,DCB=BLKSIZE=133
//GO.DDIN  DD  *
 ...schede dati...
/*
```

- Compilazione ed esecuzione di un programma PL/1 che legge delle schede dati e stampa su printer.
  Per ulteriori riferimenti circa l'uso delle schede controllo per programmi scritti in PL/1 si veda il manuale:
  CNUCE-78 : Manuale di PL/1 - parte terza

```
//PL1JOB   JOB  ...informazioni di accounting...
//        EXEC  PL1
//COMP.SYSIN DD *
 MAIN: PROC OPTIONS(MAIN);
       ....
       DCL SK CHAR(80);
       ....
       GET FILE (SYSIN) EDIT (SK) (A(80));
       PUT FILE (SYSPRINT) EDIT (SK) (A(80));
       ....
 FINE: END MAIN;
/*
//GO.SYSIN DD *
 ...schede dati...
/*
```

- Compilazione ed esecuzione di un programma Fortran che legge delle schede dati e stampa su printer.
  Per ulteriori riferimenti circa l'usc delle schede controllo per programmi scritti in Fortran si veda il manuale:
  CNUCE-50 : Programmazione Fortran sotto OS

```
//FORTJOB  JOB  ...informazioni di accounting...
//        EXEC  FORTG
//COMP.SYSIN DD *
        ....
        DIMENSION SK(20)
        ....
        READ (5,1) SK
      1 FORMAT (20A4)
        WRITE (6,2) SK
      2 FORMAT (1X,20A4)
        ....
        STOP
        END
/*
//GO.SYSIN DD *
 ...schede dati...
/*
```

- Esempio di procedura con l'uso del Linkage Editor e Loader:
Per ulteriori riferimenti circa l'uso del Linkage Editor e Loader si veda il manuale:
CNUCE-67 : Linkage Editor e Loader

```
// EXEC ....CLG
//COMP.SYSIN DD *
 ...modulo sorgente...
/*
//LKED.SYSIN DD *
 ...modulo oggetto...
/*
//GO.SYSIN DD *
 ...schede dati...
/*
```

```
// EXEC ....CG
//COMP.SYSIN DD *
 ...modulo sorgente...
/*
//GO.OBJECT DD *
 ....modulo oggetto...
/*
//GO.SYSIN DD *
 ...schede dati...
/*
```

Si noti che, con l'uso del Linkage Editor si forniscono al programma tre schede SYSIN, indicanti rispettivamente
COMP.SYSIN : i moduli sorgente che devono essere compilati;
LKED.SYSIN : eventuali moduli oggetto, gia' compilati in precedenza, che devono essere uniti, per l'esecuzione, a quelli creati in questo job nella fase di compilazione;
GO.SYSIN : eventuali dati che il programma in esecuzione dovra' elaborare.
Il Loader invece, effettuando nello stesso passo di GO anche la risoluzione degli agganci, manca dello step intermedio LKED: pertanto la scheda che definisce gli eventuali moduli in forma oggetto creati in precedenza, dovra' essere qualificata dalla parola GO. Non potendo inoltre usare la parola chiave SYSIN (GO.SYSIN indica i dati che il programma usera' in fase di esecuzione) e' stato introdotto nel sistema il riferimento alla parola OBJECT.

- Esempio di compilazioni separate: main Fortran e subroutines Assembler.
Per ulteriori riferimenti si vedano i manuali:
CNUCE-50 : Programmazione Fortran sotto OS
CNUCE-67 : Linkage Editor e Loader

```
//  EXEC      FORTGC
//COMP.SYSIN    DD  *
 ...progr. principale...
/*
//  EXEC     ASMFC
//COMP.SYSGO    DD   DISP=(MOD,PASS)
//COMP.SYSIN    DD   *
 ...subroutine 1...
/*
//  EXEC      ASMFC
//COMP.SYSGO    DD   DISP=(MOD,PASS)
//COMP.SYSIN    DD   *
 ...subroutine 2...
/*
//  EXEC      LOADER,LIB=FX
//GO.OBJECT     DD   DSN=&&LOADSET,DISP=(OLD,PASS)
//GO.FT05F001   DD   *
 ...schede dati...
/*
```

Il programma principale, in Fortran, viene compilato e messo nel data set &&LOADSET definito dalla scheda SYSLIN nella procedura catalogata FORTGC.
Successivamente viene assemblato il sottoprogramma n.1 e viene messo nello stesso data set &&LOADSET definito dalla scheda SYSGO (vedi nota a pag. 35) della procedura catalogata ASMFC.
Tale modulo oggetto si sovrapporrebbe al primo distruggendolo, se non si variasse il parametro DISP da NEW a MOD, permettendo cosi' di aggiungere il modulo oggetto relativo al sottoprogramma n.1 in coda a quello, gia' esistente, relativo al programma principale.
Analogamente opera il passo di compilazione relativo al secondo sottoprogramma.
Il passo di Loader, infine, ha come input il data set &&LOADSET definito dalla scheda OBJECT.
Si noti la mancanza dei parametri UNIT e VOL: le informazioni necessarie vengono automaticamente prelevate dallo step precedente, avendo il data set &&LOADSET la disposizione PASS.
Il Loader necessita, per creare gli agganci con i sottoprogrammi di libreria richiamati, della specifica LIB=FX.
La parola chiave LIB compare, nella procedura catalogata,

preceduta da un &: e' cioe' in forma parametrica.

```
   DSN=SYS1.&LIB.LIB
```

La codifica fornita si sostituisce a tale parametro, dando l'indicazione della libreria necessaria alla risoluzione degli agganci:

```
   DSN=SYS1.FXLIB
```

Per il corretto uso delle procedure di LKEDGO, LOADER e del parametro LIB, diamo qui di seguito la lista delle procedure:

Procedura LKED

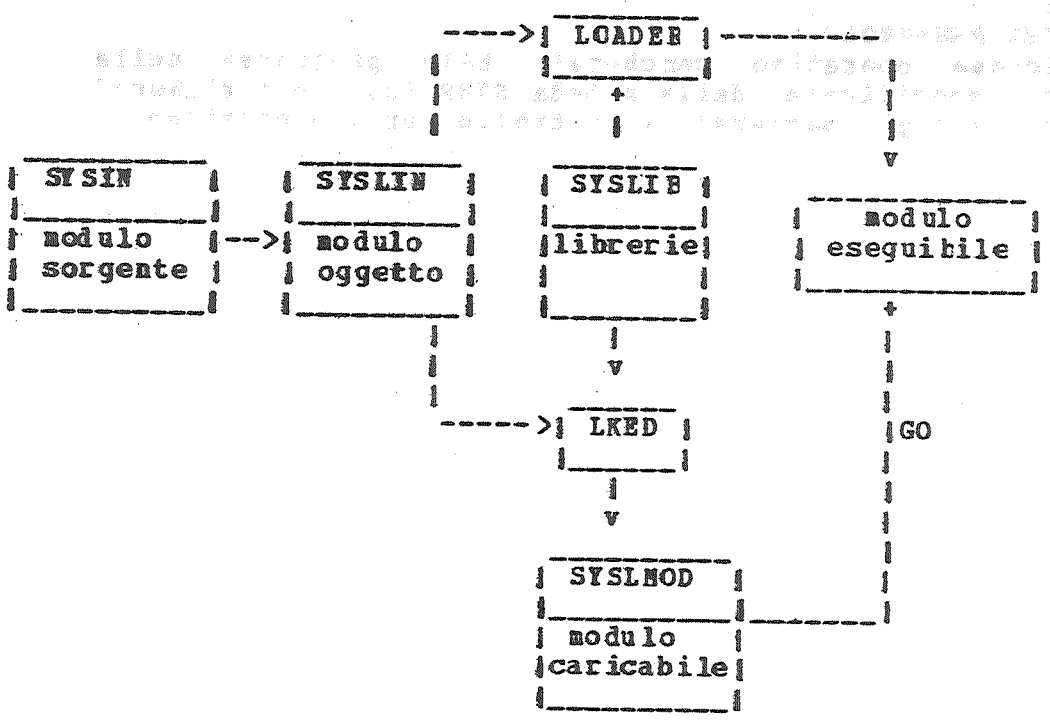
```
//****   LINKAGE EDITOR
//****   CNUCE
//* LIB=LINK PER PROGRAMMI IN ASSEMBLER
//* LIB=ALG  PER PROGRAMMI IN ALGOL
//* LIB=CB   PER PROGRAMMI IN COBOL
//* LIB=FX   PER PROGRAMMI IN FORTRAN + //GO.FT05F001 DD DDNAME=SYSIN
//* LIB=PL1  PER PROGRAMMI IN PL/1-F
//* LIB=PLR  PER PROGRAMMI IN PL/1-OPT.COMP.
//LKED   PROC  LET=LET,CALL=,OVLY=,LIB=LINK,RLKED=128K,NAME=GO
//LKED   EXEC  PGM=IEWL,PARM='MAP,LIST,&LET,&CALL,&OVLY',REGION=&RLKED
//SYSPRINT DD  SYSOUT=A
//SYSLIB DD DDNAME=FIRSTLIB
//        DD DSN=SYS1.&LIB.LIB,DISP=SHR
//        DD DSN=SYS1.ASMLIB,DISP=SHR
//        DD DSN=SYS1.LINK2,DISP=SHR
//        DD DSN=SYS1.FORT2,DISP=SHR
//        DD DSN=SYS1.PLOTLIB,DISP=SHR
//        DD DSN=SYS1.SSPLIB,DISP=SHR
//        DD DSN=SYS1.CERNLIB,DISP=SHR
//        DD DDNAME=LASTLIB
//FIRSTLIB DD DSN=SYS1.NULLPDS,DISP=SHR
//LASTLIB DD DSN=SYS1.NULLPDS,DISP=SHR
//SYSLIN    DD  DDNAME=SYSIN
//SYSUT1 DD DSN=&SYSUT1,DISP=(NEW,PASS),
// UNIT=3330,VOL=SER=SYSDD,SPACE=(13030,(133))
//SYSLMOD   DD  DSNAME=&&GOSET(&NAME),DISP=(MOD,PASS,DELETE),
//              UNIT=SYSDA,SPACE=(CYL,(3,1,1))
```

Procedura LKEDGO

```
//****   LINKAGE EDITOR AND GO
//****   CNUCE
//* LIB=LINK PER PROGRAMMI IN ASSEMBLER
//* LIB=ALG  PER PROGRAMMI IN ALGOL
//* LIB=CB   PER PROGRAMMI IN COBOL
//* LIB=FX   PER PROGRAMMI IN FORTRAN + //GO.FT05F001 DD DDNAME=SYSIN
//* LIB=PL1  PER PROGRAMMI IN PL/1-F
//* LIB=PLR  PER PROGRAMMI IN PL/1-OPT.COMP.
//LKEDGO PROC  LET=LET,CALL=,OVLY=,CONDL=4,RGO=128K,LIB=LINK,
//             RLKED=128K,NAME=GO
//LKED   EXEC  PGM=IEWL,PARM='MAP,LIST,&LET,&CALL,&OVLY',REGION=&RLKED
//SYSPRINT DD  SYSOUT=A
//SYSLIB DD DDNAME=FIRSTLIB
//        DD DSN=SYS1.&LIB.LIB,DISP=SHR
//        DD DSN=SYS1.ASMLIB,DISP=SHR
//        DD DSN=SYS1.LINK2,DISP=SHR
//        DD DSN=SYS1.FORT2,DISP=SHR
//        DD DSN=SYS1.PLOTLIB,DISP=SHR
//        DD DSN=SYS1.SSPLIB,DISP=SHR
//        DD DSN=SYS1.CERNLIB,DISP=SHR
//        DD DDNAME=LASTLIB
//FIRSTLIB DD DSN=SYS1.NULLPDS,DISP=SHR
//LASTLIB DD DSN=SYS1.NULLPDS,DISP=SHR
//SYSLIN   DD  DDNAME=SYSIN
//SYSUT1 DD DSN=&SYSUT1,DISP=(NEW,PASS),
// UNIT=3330,VOL=SER=SYSDD,SPACE=(13030,(133))
//SYSLMOD  DD  DSNAME=&&GOSET(&NAME),DISP=(MOD,PASS,DELETE),
//             UNIT=SYSDA,SPACE=(CYL,(3,1,1))
//GO     EXEC  PGM=*.LKED.SYSLMOD,REGION=&RGO,
//             COND=(&CONDL,LT,LKED)
//SYSPRINT DD  SYSOUT=A
//FT06F001 DD  SYSOUT=A
//FT07F001 DD  SYSOUT=B
```

Procedura LOADER

```
//****   LOADER
//****   CNUCE
//* LIB=LINK PER PROGRAMMI IN ASSEMBLER
//* LIB=ALG  PER PROGRAMMI IN ALGOL
//* LIB=CB   PER PROGRAMMI IN COBOL
//* LIB=FX   PER PROGRAMMI IN FORTRAN + //GO.FT05F001 DD DDNAME=SYSIN
//* LIB=PL1  PER PROGRAMMI IN PL/1-F
//* LIB=PLR  PER PROGRAMMI IN PL/1-OPT.COMP.
//LOADER PROC  LET=NOLET,MAP=MAP,CALL=CALL,RGO=128K,LIB=LINK
//GO     EXEC  PGM=LOADER,PARM=(&LET,&MAP,&CALL,'SIZE=750K'),
//             REGION=&RGO
//SYSLIB DD DDNAME=FIRSTLIB
//       DD DSN=SYS1.&LIB.LIB,DISP=SHR
//       DD DSN=SYS1.ASMLIB,DISP=SHR
//       DD DSN=SYS1.LINK2,DISP=SHR
//       DD DSN=SYS1.FORT2,DISP=SHR
//       DD DSN=SYS1.PLOTLIB,DISP=SHR
//       DD DSN=SYS1.SSPLIB,DISP=SHR
//       DD DSN=SYS1.CERNLIB,DISP=SHR
//       DD DDNAME=LASTLIB
//FIRSTLIB DD DSN=SYS1.NULLPDS,DISP=SHR
//LASTLIB DD DSN=SYS1.NULLPDS,DISP=SHR
//SYSLIN   DD  DDNAME=OBJECT
//SYSLOUT  DD  SYSOUT=A
//SYSPRINT DD  SYSOUT=A
//FT06F001 DD  SYSOUT=A
//FT07F001 DD  SYSOUT=B
```



- Esempio di creazione di una propria libreria di programmi all'interno del data set LIBNAME, creato in precedenza, ed esecuzione di un programma in tale libreria da parte di un altro job.
Per ulteriori riferimenti si veda il manuale:
CNUCE-50 : Programmazione Fortran sotto OS

```
//JOB1   JOB  ...informazioni di accounting...
//       EXEC FORTGCL
//COMP.SYSIN DD *
 ...modulo sorgente...
/*
//LKED.SYSLMOD DD DSNAME=LIBNAME(PROG1),DISP=(OLD,KEEP),
//             SPACE=,UNIT=SYSDA,VOL=SER=MYDISK

//JOB2   JOB  ...informazioni di accounting...
//       EXEC   PGM=PROG1
//STEPLIB DD  DSNAME=LIBNAME,DISP=(OLD,KEEP),
//             UNIT=SYSDA,VOL=SER=MYDISK
```

La scheda SYSLMOD fornita si sovrappone a quella esistente nella procedura catalogata, variando i valori dei parametri DSN, DISP, UNIT, VOL e SPACE in modo da far si' che il programma Fortran sia inserito, col nome PROG1, nella libreria LIBNAME esistente sul disco MYDISK.
In particolare la codifica del parametro SPACE annulla quella esistente nella procedura, cosi' da eliminare il pericolo di estensioni indebite della libreria stessa.
Nel secondo job la scheda STEPLIB definisce la libreria nella quale si trova il programma che si vuole eseguire.
Quando si richiama in esecuzione il programma con la scheda

```
// EXEC PGM=PROG1
```

il sistema operativo cerchera' tale programma nella libreria specificata dalla scheda STEPLIB, lo carichera' in memoria e gli passera' il controllo per l'esecuzione.

- E' sempre opportuno avere a disposizione una lista aggiornata delle procedure catalogate in modo da poter sfruttare a pieno i parametri e le funzioni fornite, e poter variare a piacimento le caratteristiche standard a seconda delle particolari esigenze.
  L'Utente che lo desidera puo' fare richiesta della lista delle procedure catalogate presso l'Ufficio Utenti del Centro.
  Inoltre una versione sempre aggiornata e' a disposizione per consultazione nella stanza riservata agli Utenti.

- Diamo qui di seguito la normale suddivisione dei caratteri usata nel JCL, riportando per ciascuno di essi i rispettivi nominativi:

Caratteri Alfamerici	Alfabetici Numerici	Alphabetic Numeric	da A a Z da 0 a 9
Caratteri Nazionali	Chiocciola	"At" sign	@
	Dollaro	Dollar sign	$
	Graticola	Pound sign	#
Caratteri Speciali	Virgola	Comma	,
	Punto	Period	.
	Barra	Slash	/
	Apostrofo	Apostrophe	'
	Parent. aperta	Left parenth.	(
	Parent. chiusa	Right parenth.	)
	Asterisco	Asterisk	*
	"E" commer.	Ampersand	&
	Piu'	Plus sign	+
	Meno	Hyphen	-
	Uguale	Equal sign	=
	Spazio bianco	Blank	

3c) Il sistema OS/MVS e il sistema VM/CMS

In questo capitolo si danno informazioni specifiche sull'utilizzo del sistema MVS in ambiente VM e sulle differenze con la precedente versione SVS.

Molti dei parametri a cui si fa riferimento in questo capitolo sono di uso abbastanza specifico nell'ambiente del CNUCE e servono per lo smistamento di dati e programmi tra i due sistemi MVS e VM: pertanto non sono stati descritti in precedenza nel manuale; per una descrizione piu' dettagliata e completa di tutti i parametri e' bene far riferimento direttamente al manuale IBM OS/MVS JCL.

1) INVIO DELL'OUTPUT A UNA DESTINAZIONE NON LOCALE.
(stampante VM, lettore di macchina virtuale, remoto di RSCS)

Col sistema SVS una stampa o una perforazione poteva essere inviata sul lettore di una macchina virtuale su un remoto di RSCS o in altri nodi della rete RPCNET usando una apposita codifica nel campo tra apici della scheda JOB ('commenti'). Questa interconnessione tra il sistema OS/SVS e il sistema VM del CNUCE era stata ottenuta con alcune modifiche ai sistemi HASP da un lato e RSCS dall'altro.

Con l'installazione dell'MVS lo scambio dei files di output o di input tra il sistema VS2/MVS e il sistema VM viene gestito completamente in modo standard, dal sottosistema JES2/NJE (Job Entry Subsystem/Network Job Entry) dal lato MVS e dal sottosistema RSCS NETWORKING dal lato VM. Il collegamento fisico tra i due elaboratori del CNUCE e' effettuato con un CTCA (Channel To Channel Adapter) che permette una velocita' di trasferimento di 1.5 Mbyte/sec.

Il JES2/NJE (Network Job Entry) permette di gestire una rete elaboratori connessi tramite linee BSC o CTCA, ogni singolo elaboratore diventa un nodo della rete. La rete piu' essere costituita da nodi MVS/JES2 e nodi VM/RSCS Networking. Nella installazione CNUCE l'MVS comunica direttamente solo con il VM del CNUCE tramite RSCS Networking, mentre gli altri nodi della rete RPCNET sono visti attraverso RSCS.

Il JES2/NJE permette di inviare job o files di spool da un qualunque nodo, work station o macchina virtuale facente parte della rete e di trasferire l'output a ogni nodo, work station o macchina virtuale della rete.

La destinazione default per l'output e' la locazione da dove e' stato inviato il job. Attenzione, se il job e' stato inviato da una macchina virtuale la destinazione default non e' il lettore della macchina virtuale ma una stampante o un perforatore reale del sistema VM che

possiede la macchina virtuale.

Puo' essere richiesta, per l'output, una destinazione diversa da quella default tramite JCL e schede controllo JES2. Possono essere usati i seguenti modi per specificare la destinazione di un output data set:

- Scheda /*ROUTE (con l'opzione PRINT c PUNCH): permette di specificare la destinazione dell'output a ogni nodo, work station o macchina virtuale. Tutto L'output del job, se non altrimenti specificato viene inviato alla destinazione specificata nella scheda /*ROUTE.

- Scheda DD SYSOUT: puo' essere codificatc il parametro DEST in modo da trasferire il SYSOUT data set a una particolare destinazione. Con questo parametro non puo' essere specificato il nome di una macchina virtuale o un remoto di RSCS.

- Parametri di SYSOUT: permettono di specificare, sulla scheda DD SYSOUT, un codice che fa riferimento a una scheda /*OUTPUT la quale contiene il parametro DEST che identifica la destinazione.

FORMATO DELLA SCHEDA /*ROUTE

La scheda ROUTE indica la destinazione di tutti gli output la cui destinazione non e' specificata con il parametro DEST o la /*OUTPUT e deve essere inserita dopo la scheda JOB e prima di ogni altra scheda // (JOBLIB,JOBCAT,EXEC ecc.).

La scheda ROUTE consiste nei caratteri /* in colonna 1 e 2, ROUTE in colonna 3-7, almeno un blank seguito da PRINT o PUNCH e uno o piu' blank seguiti da una destinazione. La destinazione deve essere seguita da almeno un blank il quale deve precedere la colonna 72.

Il formato della scheda /*ROUTE e' il seguente:
(sono indicati solo i parametri utilizzabili al CNUCE)

```
        | PRINT | | LOCAL      |
/*ROUTE |       | | nodo       |
        | PUNCH | | nodo.nome  |
```

PRINT
Specifica che la stampa del job deve essere trasferita alla destinazione indicata.

PUNCH
Specifica che la perforazione del job deve essere trasferita alla destinazione indicata.

LOCAL
La stampa o la perforazione devono essere eseguite in locale del nodo che ha inviato il job. Se il job e' stato inviato da una macchina virtuale, la stampa viene fatta dal sistema VM che possiede la macchina virtuale.

nodo
Nome di un nodo dove deve essere trasferita la stampa o la perforazione.
Per avere l'output su una stampatrice o perforatore collegato al sistema VM del CNUCE il nome nodo da usare e': CNUCEVM.
Per avere l'output su una stampatrice o perforatore collegato al sistema MVS il nome nodo da usare e': CNUCEMVS.

nodo.nome
Nome del nodo e nome di una macchina virtuale o remoto di RSCS dove deve essere trasferita la stampa o la perforazione.
Il nome della macchina virtuale o remoto RSCS deve essere separato dal nome del nodo da un delimitatore. Il delimitatore puo' essere un punto, due punti, una barra o una coppia di parentesi che racchiudono in nome della macchina virtuale.
L'RSCS della locazione "nodo" interpreta il campo "nome" nel seguente modo: se esiste un remoto di nome "nodo", l'output viene inoltrato su tale remoto. Se il remoto "nome" non esiste, l'output viene accodato sul lettore della macchina virtuale "nome".

Dunque, una macchina virtuale con il nome uguale a un remoto dell'RSCS locale non puo' ricevere files da RSCS.

Esempi:

/*ROUTE PRINT CNUCEVM
La stampa sara' eseguita da una stampatrice collegata al sistema VM del CNUCE.

/*ROUTE PRINT CNUCEVM.MV1
La stampa viene trasferita sul lettore della macchina virtuale MV1 definita sul sistema VM del CNUCE.

/*ROUTE PRINT CNUCEVM.REMXXX
/*ROUTE PUNCH CNUCEVM.MV2
La stampa del job viene fatta sul remoto di RSCS REMXXX mentre il punch viene trasferito sul lettore della macchina virtuale MV2.

FORMATO DELLA SCHEDA /*OUTPUT

La scheda OUTPUT specifica le caratteristiche e/o le opzioni di uno o piu' SYSOUT data set.

La scheda OUTPUT consiste nei caratteri /* in colonna 1 e 2, la parola OUTPUT nelle colonne 3-8, e da un codice che inizia da colonna 10 seguito da un blank e dai parametri. Le colonne 72-80 sono ignorate.

La scheda OUTPUT deve essere inserita dopo la scheda JOB e prima di ogni altra scheda // (JOBLIB JOBCAT EXEC ecc.).

Formato della scheda OUTPUT:
(sono descritti solo i parametri di uso piu' comune)

```
/*OUTPUT codice parametri
```

codice:
Da 1 a 4 caratteri alfanumerici, viene usato per fare riferimento a tutti i SYSOUT data set, all'interno del job, in cui il sottoparametro "form number" e' lo stesso del "codice" specificato nella scheda /*OUTPUT.
Usando un "*" come codice la scheda OUTPUT viene usata come continuazione della scheda OUTPUT precedente.

Parametri (tutti i parametri possono essere scritti in forma intera o usando l'abbreviazione di una lettera):

```
          | LOCAL
   DEST=  | nodo
   D      | nodo.nome

  ,FCB=xxxx
  ,C=xxxx

  ,FORMS=xxxx
  ,F=xxxx

  ,LINECT=nnn
  ,K=nnn

  ,UCS=xxxx
  ,T=xxxx
```

dove:

DEST=
Possono essere specificati da 1 a 4 destinazioni per ogni SYSOUT data set. Per specificare piu' di una destinazione codificare: DEST=(n[,n][,n][,n])

dove n e' una delle seguenti destinazioni:

LOCAL
  Indica una stampatrice o un perforatore locale del nodo da cui e' stato inviato il job.

nodo
  Nome del nodo dove deve essere trasferito il SYSOUT data set. Per trasferire un SYSOUT data set al sistema VM del CNUCE il nome del nodo da usare e': CNUCEVM.
  Per ottenere una stampa o perforazione su una stampante o perforatore del sistema MVS il nome nodo da usare e': CNUCEMVS.

nodo.nome
  Nome del nodo e nome della macchina virtuale o remoto di RSCS a cui deve essere trasferito il SYSOUT data set.
  Per l'interpretazione dal campo "nome" da parte di RSCS Networkimg vale quanto si e' detto sotto la scheda /*ROUTE.

FCB=
  Da 1 a 4 caratteri alfanumerici che indicano il form control block da usare.

FORMS=
  Un valore alfanumerico che indica il FORM di stampa o perforazione (da 1 a 4 caratteri); cfr. il parametro "carta" sulla scheda JOB.

LINECT=
  Un valore da 0 a 255 che specifica il numero di linee che devono essere stampate per pagina; cfr. il parametro "linee" sulla scheda JOB.

UCS=
  Da 1 a 4 caratteri alfanumerici che indicano l'UCS da usare.

Nota
  I parametri codificati su una scheda OUTPUT rimpiazzano ogni equivalente parametro specificato sulla scheda DD referenziata.
  Possono essere usate tante schede OUTPUT quante sono necessarie. Se piu' di una scheda OUTPUT ha lo stesso codice a colonna 10 sara' usata la prima.

Esempi:

```
//PROVA JOB ......
/*OUTPUT ABCD DEST=CNUCEVM
// EXEC ....
//STAMPA DD SYSOUT=(S,,ABCD)
//STAMPA1 DD SYSOUT=A
/*
```

Nell'esempio precedente l'output creato sul ddname STAMPA verra stampato da una stampatrice collegata al sistema VM e di classe "S" (stampante self-service). Le altre stampe (jcl e altri sysout) saranno stampati in locale dal sistema VM.

```
//PROVA JOB ......
/*ROUTE PRINT CNUCEMVS
/*OUTPUT AAAA DEST=CNUCEVM.VM1
/*OUTPUT BBBB DEST=CNUCEVM.REMXXX
// EXEC ....
//STAMPA1 DD SYSOUT=(A,,AAAA)
//STAMPA2 DD SYSOUT=(A,,BBBB)
//SYSPRINT SYSOUT=A
/*
```

Nell'esempio precedente l'output creato sul ddname STAMPA1 verra' trasferito sul lettore delle macchina virtuale VM1. L'output creato sul ddname STAMPA2 verra' stampato sul remoto di RSCS di nome REMXXX. Le altre stampe (jcl e altri sysout) saranno stampati in locale dal sistema MVS.

```
//PROVA JOB ......
/*ROUTE PRINT CNUCEVM.VM1
/*OUTPUT AAAA DEST=(CNUCEVM.VM2,CNUCEVM.VM3)
// EXEC ....
//STAMPA1 DD SYSOUT=(A,,AAAA)
//STAMPA2 DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
/*
```

Nell'esempio precedente l'output creato sul ddname STAMPA1 verra' trasferito sui lettori delle macchine virtuali VM2 e VM3. Le altre stampe (jcl e altri sysout) saranno trasferite sul lettore della macchina virtuale VM1.

Quando un file di stampa o perforazione viene trasferito da MVS a VM la classe di sysout viene conservata. Questa proprieta' puo' essere utilizzata per ottenere la stampa di un job sulla stampante self-service.

Esempio.

```
//... JOB ....,MSGCLASS=S
/*ROUTE PRINT CNUCEVM
// EXEC ...
...
//xxxx DD SYSOUT=*
```

Sulla scheda JOB deve essere codificatc il parametro MSGCLASS e tutte le SYSOUT devono essere di classe S o * (come nell'esempio). La codifica SYSOUT=* significa che la classe di sysout e' uguale al MSGCLASS.
Attenzione le sole classi di sysout utilizzabili in MVS per file di stampa sono A e S.

Un altro metodo per specificare la classe di una stampa, dopo il trasferimento al VM, consiste nell'indicare un form number di un solo carattere sulla scheda JOB. Dopo il trasferimento della stampa al VM il form number diventa la classe di spool. Si ricorda che il form number e' il sesto parametro tra parentesi sulla scheda JOB.

```
//... JOB (0000,xxxx,1,1,,2)
/*ROUTE PRINT CNUCEVM
```

L'output del job viene trasferito al VM e stampato da una stampatrice di classe 2.

2) INVIO DI UN JOB DA UNA MACCHINA VIRTUALE.

Per l'invio di un job al sistema MVS da una macchina virtuale si puo' continuare ad usare il comando VS messo a disposizione dal CNUCE. Se pero' si vogliono usare i comandi standard di VM e' necessario codificare la parola JOB nel comando TAG.

Il comando TAG deve essere del tipo:

```
TAG DEV 00D CNUCEMVS JOB
```

La parola JOB indica che il file deve essere trattato come un job in input al sistema MVS.

Se non si specifica la parola JOB, nel comando TAG, il file viene trattato come un file di spool, trasferito al sistema MVS e immediatamente messo in attesa di perforazione.

Esempio:

Supponiamo di dover inviare un job, le cui schede controllo si trovano nel file PROVA VS e i dati si trovano nel file DATI DATI, i comandi da usare sono allora i seguenti:

```
CP SPOOL 00D RSCSNET CONT
CP TAG DEV 00D CNUCEMVS JOB
PUNCH PROVA VS (NOH
PUNCH DATI DATI (NOH
CP SPOOL 00D NOCONT CLOSE
CP SPOOL 00D OFF
```

3) TRASFERIMENTO DIRETTO DI FILE DA READER A PRINTER O PUNCH (EX SCHEDA SEND)

La scheda SEND, che richiedeva una modifica al sistema, non e' piu' utilizzabile in MVS.
Una funzione equivalente puo' essere ottenuta in maniera standard utilizzando il parametro TYPRUN=COPY della scheda job.

Esempio:

Si vuole trasferire un pacco di schede dal lettore collegato all'MVS, ad una macchina virtuale in testa al pacco di schede devono essere inserite le due schede seguenti:

```
//... job (...   ),TYPRUN=COPY,MSGCLASS=B
/*ROUTE PUNCH CNUCEVM.nomemv
....
....
```

La scheda job deve essere valida per l'MVS. In questo esempio si e' utilizzato il parametro MSGCLASS=B perche' il file trasferito deve essere di tipo punch.
Sono trasferite tutte le schede compresa la JOB e la ROUTE.

4) UTILIZZO DELL'INTERNAL READER.

La specifica UNIT=INTRDR, valida in SVS, e' errata in MVS; per utilizzare l'internal reader la corrispondente scheda DD deve essere codificata nel modo seguente.

```
//nome DD SYSOUT=(A,INTRDR)
```

5) ORGANIZZAZIONE VSAM.

Rispetto ai metodi di accesso ai dati forniti dal sistema OS/SVS (sequenziale, partitioned, index-sequential), il sistema MVS rende disponibile il metodo di accesso VTAM.
Le caratteristiche operative del nuovo metodo di accesso sono state progettate per la soluzione sia di problemi batch, in cui il VSAM si distingue per l'ottimizzazione dello spazio disco e per la flessibilita' di utilizzo, sia di problemi di teleprocessing, per i quali esso assicura un accesso accurato e rapido.
Caratteristiche fondamentali del VSAM sono un alto grado di affidabilita' e sicurezza e le elevate prestazioni, normalmente superiori a quelle dei metodi di accesso precedenti, che potranno quindi essere vantaggiosamente sostituiti.

6) UTILIZZO DEL CATALOGO

In MVS il catalogo principale e' un catalogo VSAM. In esso sono catalogati solo i data set di sistema e i data set necessari a tutti gli utenti.
Il catalogo principale, contenendo dati di vitale importanza per il sistema, e' protetto contro accessi non autorizzati, di conseguenza un utente non puo' catalogare un data set al suo interno.

Consigliamo gli utenti a non usare, nei limiti del possibile, data set catalogati. In ogni caso per chi desidera utilizzare il catalogo e' stato creato un apposito catalogo (user catalog VSAM) di nome USER.

Per catalogare un data set, con il parametro DISP=(NEW,CATLG), una delle due condizioni seguenti deve essere soddisfatta:

a) Il nome del data set dave avere come primo qualificatore il nome del catalogo o un suo alias (es. USER.DATASET.MIO).

b) Deve essere presente una scheda JOBCAT o STEPCAT che identifica il catalogo.

Se nessuna di queste condizioni e' soddisfatta si verifica una condizione di errore e il data set non viene catalogato.

Esempi:

a) Uso della scheda JOBCAT.

```
//.... JOB ...
//JOBCAT DD DSN=USER,DISP=SHR
//     EXEC ....
//xxx  DD DSN=NOME,DISP=(NEW,CATLG),....
...
```

b) Uso del primo qualificatore uguale al nome catalogo.

```
//.... JOB ...
//  EXEC ....
//xxx DD DSN=USER.NOME,DISP=(NEW,CATLG)
...
```

L'uso del primo qualificatore del nome data set uguale al nome del catalogo impedisce di usare l'autorizzazione automatica dei data set, per cui si consiglia di utilizzare le schede JOBCAT o STEPCAT.

La ricerca di un data set catalogato viene fatta dal sistema nel seguente modo:

a) La ricerca avviene nel user catalog specificato nello step corrente (con una scheda STEPCAT DD). Se piu' di un catalogo e' specificato per lo step, la ricerca avviene nell'ordine di concatenazione. Se il data set viene trovato non viene fatta nessuna altra ricerca; se invece il data set non viene trovato, un eventuale catalogo individuato con una JOBCAT non viene preso in considerazione e la ricerca continua al passo b.

   Se nessun catalogo e' specificato con una STEPCAT, e un user catalog e' specificato per il JOB (con una scheda JOBCAT DD) viene controllato il catalogo individuato dalla JOBCAT. Se il data set viene trovato non viene fatta nessuna altra ricerca. Altrimenti si prosegue col passo b.

b) Se il data set ha un nome qualificato e il primo qualificatore e' lo stesso del:

   nome di un user catalog, o

   un alias di un user catalog, o

   un alias di un control volume,

   la ricerca avviene nell'user catalog o nel control volume individuato. Se il data set viene trovato non viene fatta nessuna altra ricerca. Altrimenti si prosegue col passo c.

c) La ricerca avviene nel catalogo principale. Se il nome del data set non viene trovato la ricerca fallisce.

Per scatalogare un data set valgono le stesse considerazioni della catalogazione.

Essendo i cataloghi in MVS strutture VSAM per la loro gestione, catalogazione, scatalogazione, lista, deve essere usato il programma di utilita' ACCESS METHOD SERVICE. I programmi di utilita' IEHPROGM e IEHLIST non possono essere piu' usati per la gestione dei cataloghi.