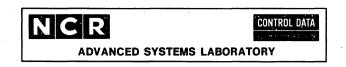
ADVANCED SYSTEM LABORATORY

CHP0604

1

IPLOS GDS - PROGRAM MANAGEMENT



CHAPTER OL

PROGRAM MANAGEMENT

Doc. No. ASLOD282

Rev. 04

Copy No. 87



REV 29 APR 75

TABLE OF CONTENTS

																																•	4
4	.0		т	NT	0.0	.		-																									5
	•1											<u> </u>		•				•	•						•		٠	•	٠	٠	٠	1-1	
	.2												OE												•	٠	•	•	•	٠	٠	1-4	7
Ŧ	• ٢		U	Er	Τr	11	11	UN	1 (J٢	1	E۲	RWS	Ś	•	•	•	•	•	٠	٠	٠	٠	٠	•	٠	٠	٠	٠	٠	٠	1-4	8
~	~		_																														9
	• 0	_									TI			•		•			•	٠	٠	٠	٠	•	٠	٠	٠	•	٠	•	•	2-1	10
2							N	CC	IN:	51	RU	Ç	S.		•	•			•	•	٠	٠	•	٠	•	٠	٠	٠	•	٠	٠	2-1	11
			•1					٠	٠	- •	٠	•	•	•	٠	•	•	•	•	•	٠	٠	٠		•	•	٠	•	٠	•	٠	2-1	12
			• 2					•	٠	٠	٠		•	•	•	•	•	•	•	•	•	٠	•		٠		٠	•	•	•		2-2	13
			• 3							٠	٠	•	•	•	•	•	•	•	•	•	٠	•	•	•	•	•	٠		٠	٠	•	2-2	14
2	• 2									IM	ΕN	Т			•					•	٠	٠	•	٠	•	•	٠	•	•	•	•	2-3	15
			•1							٠	•	•	•	•	•			•	•	•	•	٠	•	٠	•	٠	•			•		2-3	16
	2.		• 2						•		•		•				•		•	•	•	٠	•	•								2-4	17
		2	•2	• 2	• 1	. 1	'n	o g	ir a	a m	С	or	tr	01	8	10)cł	۲		•	•	•	•	•	•	•						2-4	18
													1						•	• .	•	•					۰.					2-5	19
		2	• 2	• 2	• 3		Ēs	ta	bl	1	sh	еc	I P	ro	gr	an	10	Co	nt	ra	51	в	l o	ck								2-6	20
		2	•2	•2	•4	•	Jo	b	Ga	t i	e	Тa	ьI	e -					•	•												2-6	21
		2	• 2	• 2	• 5	,	Jo	b	S 1	a	ck	T	аb	l e		•				•	•	•										2-7	22
	2,	• 2	• 3	Т	AS	К	Ε	ST	AB	۶L	ΙS	ΗМ	EN	T	ΕX	AM	PL	E.		•	•	•		•								2-7	23
	2.												SH						•	•												2-13	24
		2	•2	•4	. 1	5	su	bt	as	ĸ	С	on	tr	o I	в	10	ck	(•								1				2-13	25
2,	.3	Ρ	RO	GR	AM	8	ΞX	EC	UT	I	ΟN	R	EQ	UE	ST	s																2-14	26
			•1						ΤE		•			•																		2-14	27
	2.	3	• 2	Ρ	M#	E)	(I	Т		•	•		÷							•		•										2-15	28
	2.	3	• 3	Ρ	M#	TE	R	MI	NA	T	E	•	•																			2-15	29
			•4							•	•	•																				2-16	30
	2.	. 3	• 5	P	1#	LC	A	D		•							۰.															2-16	31
			• 6								•			•																		2-16	32
	2.	3	• 7	Ρ	M#	RE	Ι	NI	ΤI	A	LI	ΖE																				2-17	33
			•8																													2-17	
	2.	3.	• 9	PI	1#	01	S	ES	ΤA	8	.1:	SН																				2-18	35
																						-	-	-	-	•	-	•	•	•	•	- 10	36
3.	0	L	DG :	IC/	۱L	٨	IA I	ME	S	P/	ACE	Ξ	MA	NA	GΕ	ME	NT															3-1	37
	3.	٥.	. 1	D	ΞS	ΙG	N	0	ΒJ	E	СΤС	Ľ۷	ES																			3-2	38
з.	1	S١	1 S 1	ΓEΙ	1	DE	S	CR	ΙP	Τ.	C O I	1																				3-2	39
	З.	1.	1	LI	٩S	D	E	SCI	٦I	P1	ror	٢S										-										3-4	40
	3.	1.	2	LN	IS	D	A 1	ΓA	T	YF	PE S	5																				3-6	41
	3.	1.	.3	L	٩S	S	T	RUI	СТ	UF	RES	S																				3-7	42
		3.	.1 .	.3.	1	G	er	n er	۰a	L	E	a	np	le	5	-				-			-		-				•			3-7	43
		3.	1.	3.	2	S	Сι	#	τо	KE	ΞN	Ε	xaı	סת	le														:	•	:	3-11	43
	З.	1.	4	T١	P	Ξ	С	DN1	r R	οL	LE	D	••(ואכ	1	20	nĒ	•• `	P	20	CF.	ňIJ			•			•	•	•	•	3-13	44
	3.	1.	5	E)	(T i	RI	NS	SIC	5	A 1	TF	RI	30	TE:	s									•			:	•	•	1	•	3-13	45
3.	2	LN	1S	RE	Q	JE	SI	T S					•													:		•	•	•	•		
	3.														1							•	•	:	•	:		•	•	•	•	3-14	
	3.	2.	2	LN	IS	¢D	ΕI	r A (ЭН				:	:									•		•	•	•	•	•	•	•	3-15	48
	3.	2	3	LN	IS I	۶Ó	Ēſ	L	R	ΕÏ	7												•	•	:	:	:	:	:	•	•	3-16	49
	з.	2.	4	LN	ISI	ŧR	Ē١	10	/E			Ĩ		:			-						•	•	•	•	•	•	•	•	•		50
	з.	2.	5	LN	IS	ŧE	NI	R	(•	•	:	•	:	:	•	•	•	•	•	3-18	51
	3.												:					•	1			•	•	•	•		:	٠	•	•	•	3-19	52
	3.									•						:	•	•	•		•	•	•	•	•	-	-	•	•	•	•	3-20	53
							- 4		•		•	•	•	•	•	•	•	•	•	•	•	•	•	٠	•	•	•	٠	٠	•	•	3-21	54

3.2.8 LNS#GROW	3-22	1
	3-23	2
3.2.10 LNS#UNLOCK	3-24	3
3.2.11 LNS#INSERT	3-25	4
	3-26	5
3.2.13 LNS#GET	3-27	6
	3-28	7
3.2.15 LNS#SETXA	3-29	8
	3-30	9
	3-31	10
	3-33	11
3.3.3 LNS#SEGLOCK	3-34	12
	3-35	13
	3-35	14
	3-36	15
	3-37	16
		17
4.0 PROGRAM COMMUNICATIONS	4-1	18
4.1 EVENTS	4-1	19
4.1.1 EVENT REQUESTS	4-3	20
4.1.1.1 PM#ATTACH_PROCEDURE	4-3	21
4.1.1.2 PM#CAUSE_EVENT	4-3	22
4.1.1.3 PM#CAUSE_CLEAR_EVENT	4-4	23
4.1.1.4 PM#CLEAR_EVENT	4-4	24
4.1.1.5 PM#DETACH_PROCEDURE	4-4	25
4•1•1•6 PM#DISABLE_EVENT • • • • • • • • • • • • • • • • • • •	4-5	26
4.1.1.7 PM#ENABLE_EVENT	4-5	27
4.1.1.8 PM#STATUS_EVENT	4-5	28
4.1.1.9 PM#WAIT_EVENT	4-6	29
4.1.1.10 PM#WAIT_CLEAR_EVENT	4-7	30
4.2 SIGNALS	4-8	31
4.2.1 SIGNAL SELECTION		32
4.2.2 SIGNAL REQUESTS	4-9	33
		34
4.2.2.2 PM#SELECT_SIGNAL		35
		36
		37
		38
	4-11	39
		40
		41
		42
	• = •	43
		44
		45
		46
		47
		48
		49
		50
4.4.3 INTER-JOB SYNCHRONIZATION		51
		52
4.4.3.1.1 PM#SIGN_LOCK		53
4.4.3.1.2 PM#UNSIGN_LOCK	4-19	54

A-2 75/05/21

A-3 75/05/21

0	P	ROO	RAM	MA	IN	TEN	A N	CE		•	٠	٠	•	•	•	•	•	•	٠	•	•	•	•	•	•	5-1	
1	003	0L	LOC LOC	ΚN	OT	ES	т •	•	TE: •	s •	•	•	•	•	•	•	•	•	•	•	•		•	•	•	6-1 6-1 6-4	
					÷																						
																								•			
																						÷					

CHP0604 ADVANCED SYSTEM LABORATORY 75/05/21

IPLOS GDS - PROGRAM MANAGEMENT -----

1.0 INTRODUCTION

1.0 INTRODUCTION

IPLOS Program Management provides the mechanisms through which the user may organize and present his programs to the system. The three basic constructs of Program Management are:

STATIC CONSTRUCT	DYNAMIC CONSTRUCT	CHARACTERISTICS	ANALOGIES	1
Job	I Job I I I	I.Single address space .Batch submission or single user terminal session	Job in most systems	1
Program	Task I I I	I.Separate naming context (entry pts - externals) .Separate common block allocations .Separate load	PLJS Task	-+ 1 1 1 1
Procedure	I Subtask I I		PL/I Task CENTURY B2 Task BURROUGHS Async Procedure	-+ 1 1 1

TABLE 1.0-1

PROGRAM MANAGEMENT BASIC EXECUTION CONSTRUCTS

The progression from job to task to subtask is characterized 41 by a.) decreasing amounts of static data, b.) decreasing overhead involved in initiation, and c.) increasing amounts of automatically shared data.

Each of these constructs is dealt with in greater detail in the ensuing parts of this section.

NCR/CDC PRIVATE REV 29 APR 75

ADVANCED SYSTEM LABORATORY

1-1

1

2

3

4

5

8

q

10

11

12

36

37

38

39

40

42

43

44 45

46

47

48

CHP0604

75/05/21

1-2

IPLOS GDS - PROGRAM MANAGEMENT ~~~~

1.0 INTRODUCTION

Program Management also provides the mechanisms for communications between jobs and between programs in execution.

For communication between nonsimultaneously active jobs, a mailbox file is provided. The mailbox provides a permanent repository (i.e., unrelated to the life of a particular job), for messages. This enables jobs to enter the system in arbitrary order, at arbitrary times, and to sequence and synchronize their subsequent activations.

For executing jobs, tasks, or subtasks, the following communication mechanisms are available:

> LNS Signals Queues Events Semaphores Signature locks On conditions

These mechanisms allow lobs, tasks, and subtasks to synchronize and coordinate themselves with other asynchronous activities. These mechanisms and the requests which are used to manipulate them are treated in greater detail in ensuing parts of this section.

1

2

	SYSTEM LAE	1 MANAGEME	ENT	CHPO	75/05/21	ADVANCED SYSTEM LABORATORY CHP0604 75/05/21 IPLOS GDS - PROGRAM MANAGEMENT
1	.0 INTROD					1.0 INTRODUCTION 1.1 REQUIREMENTS AND OBJECTIVES
	I TYPE	SCOPE	DATA	LIFETIME		1 1.1 <u>REQUIREMENTS AND OBJECTIVES</u>
	1 .	Job I		8	I Job sequencing, I I Communication I I between users I	The following is a summary of the major requirements and bjectives that motivate the design of IPLOS Program Management:
	l local LNS global	Intra Job	Predefined by type	Job	I Symbolic access I from terminal, Passing parameters from user to the system I	8 Asynchronous Processing Facility) 9 0 9 0 10 0 11 0 12 0 13 10 14 0 15 0
		Job I		Stack, Static	Synchronization, Interrupt control 	16 17 1.2 <u>DEFINITION OF TERMS</u> 18 19
	Signal 	Inter I	128 bytes	DEQUEUE or Overwrite	Communications	20 The following are definitions of terms relevant to 'Program 21 Management. 22 23
·	1 1	Job 🛔	1	(Job) LNS,	t Queuing signals, I I passing data I	24 Address Space The set of segments addressable in a 25 Job. Each address is uniquely identified 26 by a segment number and a byte number. 27
	Sema- phore 	Intra Job	Integer	(Job) LNS, Stack, Static	<pre>Synchronization, 1 Locking (using 1 shared resources) 1</pre>	28Binary Object FileA file containing one or more contiguous29object modules. All object modules in30the segment have the same segment31attributes.32
	Sign Locks	Inter Job	Compare Swap word	Segment	I Synchronization I I (Compare Swap) I	32 33 Binding Section. The object environment component used to 34 control transfer between rings of 35 protection. There is one binding section
	I dition I I I	Program1	Register		Handling execution condition. See 1 Doc ASL00211.	36 per loaded module. 37 38 Binding Segment A segment containing the binding sections 39 of one or more modules loaded into the
	•		TABLE			40addressspaceofajob.Thereare41several binding segments per job.4243ConditionA synchronous occurrence of interestto43ConditionA synchronous occurrence of interestto44thetaskorsubtaskin45occurred.The arithmetic faults, such as46overflow, are examples of conditions.
						47 48 Control Point The basic execution entity recognized and

NCR/CDC PRIVATE REV 29 APR 75

NCR/CDC PRIVATE REV 29 APR 75

VANCED SYSTEM LABORATORY	CHP 0604		ADVANCED SYSTEM LABORATORY	1-6 CHP 0604
LOS GDS - PROGRAM MANAGEMEN	75/05/2 T		IPLOS GDS - PROGRAM MANAGEMENT	75/05/21
1.0 INTRODUCTION 1.2 DEFINITION OF TE		~-	1.0 INTRODUCTION 1.2 DEFINITION OF TERM	
	dispatched by the System Monitor。 Amo its contents is the hardware defin Exchange Package.		Job Gate Table	A table used by Program Management to register gated entry points on a job basis.
Control Point id	A system unique identification of control point used as the destinati address of signals.		Job Stack Table	A table used by Program Management for ring by ring allocation of stacks when a control point is created.
Entry Point Event	A named externally accessible address a module. The entry points may be either the code section or the worki storage section of the module. An asynchronous occurrence	in 10	Library	A segment containing procedures and the dictionaries required to locate them. The procedure dictionary is organized by entry point name. All load modules in the library have the same segment attributes.
Event Control Block	significance to a task or subtask. Ta completion, time, and I/O completion a typical examples of events. A data structure required to manipula	re 16 17 18	Load Module	An object module reformatted by OBLIGE for residency on a library. Can be a single procedure. Structured as directly
	the flow of control via event request: May be in LNS, internal static, or stack.	s. 20	Loader Map	referenceable storage; code section shareable among users. The output of the Loader describing the allocations performed for all the
External	A symbol referenced by a module that defined as an entry point in anothe module.	is 24 er 25 26		sections of all the modules in the loaded program.
Gate	A hardware protected entry point for crossing between programs. Protection changes can only occur at gates Validation of the right to change can	on 29 s. 30	Loader Symbol Table	An internal table built and used by the Loader for matching externals and entry points. There is a separate Loader Symbol Table per loaded program.
Gate Registration	done at the gate. The act of making a gate known within job, such that subsequent loading wi link to the protected entry point who	11 35	Local Key	One of the two keys associated with every known segment. Verified on every access. Always associated with the segment and not verified or passed on by call/return sequences.
Global Key	referenced. One of the two keys associated with even known segment. Verified on every acces	ss 40	Mailbox	A file used for communication between two users, for example, for job sequencing. May contain messages.
	and on call/return sequences. Intende as a mechanism for isolating program executing in the same ring protection. Not supported in v 1.0.	ms 42	Object Module	A single place of machine executable code output from a compiler. Structured as a series of records on a file that are interpreted every time the object module
Job	Job is defined in Section 1.0 of Chapte 4 of the OSGDS.		Procedure	is processed. Code that may be executed serially via
	NCR/CDC PRIVATE REV 29 APR 75			NCR/CDC PRIVATE REV 29 APR 75

. •

ADVANCED SYSTEM LABORATORY	CHP0604	1-7	ADVANCED S	YSTEM LABORATORY		CHP0604	1-8 75/05/21
IPLOS GDS - PROGRAM MANAGEMENT		75/05/21	IPLOS GDS	- PROGRAM MANAGEM	1ENT		
1.0 INTRODUCTION 1.2 DEFINITION OF TER	MS	·····		INTRODUCTION DEFINITION OF	TERMS		
	hardware call instruction o asynchronously via spawning a				jobs.		x
Program	A set of object files, set of and an entry point name which static set of procedures o perform some specific funct compile COBOL statements). An	3 libraries, 4 specifies a 5 rganized to 6 ion (e.g., 7		bsystem	in the sam System Job	n provides service me way as those pr . It is protect d the Operation from it.	rovided by the ted from the
Program Control Block	of a program is a task. LNS structure required to c program by linking external and entry points in a specif It can be in any LNS segment.	9 10 onstruct a 11 references 12	Su Su	bsystem Services	internal s services ar have the scheduling as the requ	hared procedures static) which prov nd are directly ca e same clock and execution cl uestor. The only ss rights to da	vide Subsystem allable. They accounting, naracteristics difference is
Queue	A collection of data item processing. Standard sig queued.	s awaiting 16 nals are 17 18			They are Services, 1	also protecte that is, in a dif	ed from Task ferent ring.
Queue Control Block Ring	A data structure required to m queue via queue requests. May internal static, or a stack.	be in LNS, 21 22 23		btask	within a associated with the s only a new	us execution of single task. A with the task subtasks. The sub stack segment as	II static data is associated otask receives
KING	The fifteen hierarchial protection available within job. Used to protect local m services from their users. Ca	onitors and 26	Та	sk	for private Identifiab	e data. le execution of a	program.
	ring n is always greater than capability in ring n+1.		Та	sk Control Bloc		LNS data structur task and pass it	
Semaphore	A system supported facility synchronization among a activities within a job. It is primitive such facility suppor system.	to permit 31 synchronous 32 s the most 33	Ta	sk Monitor	reentrant	tion of shared, procedures which formal interface monitor.	n monitor and
Signal	A signal is a short message used for inter-job communicat form of requests and responses	lons in the 38		sk Services	internal si System s callable.	hared procedures tatic) which prov services and a They have the , scheduling a	ide Operating are directly same clock
Signal Buffer	A system structure used to signal reception by a control	interface 41			characteri	stics as the re rence is their ac	questor. The
Signal Selection List	t A system table used to regist selections on a control point f	ster signal 44					
Signature Lock	The externalization of Compart locking data in shared segmen	e-Swap for 47					
	NCR/CDC PRIVATE REV 2	29 APR 75			N	CR/CDC PRIVATE R	EV 30 APR 75

ADVANCED SYSTEM LABORATORY	CHP0 60 4	2-1
IPLOS GDS - PROGRAM MANAGEMENT		75/05/21
2.0 PROGRAM EXECUTION		
2.0 <u>PROGRAM EXECUTION</u>	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	
		2
		5 5
		6
2.1 EXECUTION CONSTRUCTS	*	8

IPLOS supports three major execution constructs:

- o JOB
- O TASK
- O SUBTASK

2.1.1 JOB

The job is the mechanism through which the batch or interactive user interfaces to the IPL system. A job consists of a <u>single</u> segmented address space and all the work performed by the job takes place within that address space.

The convention of associating a single address space with a job is not mandatory, however, the OS project feels that there are several factors which make it desirable:

- o It allows natural sharing of information between components of the job - all information is addressed 31 through the same mechanism (i.e., the same segment descriptor table)
- o It allows the code which manages the components of a lob -35 (i.e., program establisher, task establisher, loader) to 36 be a part of the same job thereby a.) facilitating the 37 component management and b.) isolating it from other jobs 38 and the system code responsible for job management. 39
- o It allows large amounts of the system and user provided 41 environment that all components of the job depend upon to 42 only be established once for all the components in the job 43 (e.g., task monitor, subsystem services, etc.) 44
- o It allows straightforward invocation and parameter passing 46 between the aforementioned shared environment and a user 47 task. 48

NCR/COC PRIVATE REV 30 APR 75

DVANCED SYSTEM LABORATORY

CHP0604

2-2

5

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24 25

26

27

28 29

75/05/21

IPLOS GDS - PROGRAM MANAGEMENT

- 2.0 PROGRAM EXECUTION
- 2.1.1 JOB

There are also several disadvantages to the single job per address space relationship:

- o The volatility of comings and goings of programs and data within the address space forces the loading of absolutized components to be preplanned (i.e., the permanent reservation of a segment in every address space).
- o Components that are independent of each other and have therefore no need to share or communicate are unprotected from each other and therefore subject to time dependent errors. This may be improved somewhat by utilizing the various protection mechanisms available within the the address space (.e.g., rings, global or local keys).

In spite of these disadvantages, we feel that the single address space per job is the best way to proceed.

2.1.2 TASK

10

11

12

13

14

15

16 17

18

19

20

21

22

23

24

25

26

27

28

29

30

32

33

34

40

45

A program is the principal way work is organized for the user by Program Management. It is the typical unit of loading and execution. The program itself is a static entity, that is, it is the object files and libraries which get established and linked for each separate execution of the program. Each one of those executions is a separate task.

Each task represents a separate loading and execution 30 environment. Any common blocks (i.e., FORTRAN common, PL/I 31 static external, COBOL' global) declared in the task are 32 accessible by any procedure in the task. All entry 33 point - external reference matchings with the exception of gate 34 linkages are evaluated in the task context. No data is 35 automatically shared between tasks in the same job, however, 36 since they are in the same address space, sharing segments is 37 facilitated.

2.1.3 SUBTASK

A procedure is a logically discrete piece of code that is 44 the basic component of a program. A procedure may be compiled 45 with other procedures to form a single object module; may be 46 bound by the library generator with other object modules, or may 47 be linked to other discrete object modules at execution time. 48

ADVANCED SYSTEM LABORATORY	2-3	
		75/05/21
IPLOS GDS - PROGRAM MANAGEMENT		~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
2.0 PROGRAM EXECUTION		
2.1.3 SUBTASK		

A subtask is an asynchronous activation of a procedure. procedure whether called as a "subroutine" within a program or called asynchronously has a single allocation of data associated with in at call time. The data is the variables that are local to the procedure in the block structured language sense (e.g., PL/1 automatic). The allocation is made at call time in the run time stack segment provided by Program Management. In the case of the spawning of a subtask, a new stack segment is provided for the subtask for its stack frame and the stack frames of any procedures serially called by the subtask. This is the only 10 private data associated with the subtask. All static data and 11 linkages associated with the spawner are associated with the 12 spawned subtask as well. A subtask is intended to be the most 13 efficiently established asynchronous facility supported by 14 Program Management. This will be effected by only providing it 15 with the minimum necessary amount of environment. 16

2.2 TASK_ESTABLISHMENT

A task is defined to Program Management with a Task Control Block (TCB). The TCB specifies the program to be executed and its execution environment. A task is established by issuing a PM#EXECUTE request. Task establishment consists of loading a program, and creating a control point with an exchange package 26 and stacks for established programs in different rings that will be called during the course of execution. The simplest task 28 example would be one with an exchange package, a stack for the 29 user program, and a stack for the task services program. 30

Subsystem Services programs can be established and included in the execution environment. A control point and stacks are created by a PM#EXECUTE request but not by a PM#ESTABLISH request. Both effect the loading of a specified program.

2.2.1 LOADING

A program is defined to Program Management with a Program 41 Control Block (PCB). The PCB specifies a list of object files, a 42 list of library files, and an entry point for the program. The 43 Loader uses this information to construct an object module 44 segment, a working storage segment, and a binding segment. 45 46

First the Loader builds the object module segment from the 47 list of object files, if specified. An object file is generated 48

NCR/CDC PRIVATE REV 30 APR 75

ADVANCED SYSTEM LABORATORY

2 - 7

1

2

3

4

5

6

7

8

9

17

18

19

20

21

22

23

24

25

27

31

32

33

34

35 36

37

38

39

40

2-4 75/05/21

IPLOS GDS - PROGRAM MANAGEMENT

2.0 PROGRAM EXECUTION

2.2.1 LOADING

by a compiler and may contain one or more object modules that represent code in a nonexecutable form. The format is detailed in Chapter 11 of the OSGDS. For each object module, the Loader creates an executable code section in the object module segment, a working storage section and a binding section.

CHP0604

Next the Loader resolves unsatisfied externals using the library segments. The listed library segments are represented (via SC#INITIATE_SEGMENT) in the address space of a job as is, with one process segment per library. File attachments must be done prior to this step. A library segment contains an entry point dictionary and one or more load modules. The difference between a load module and an object module is that the code section of a load module is already in executable form. For each referenced load module, the Loader generates a working storage section and a binding section in the corresponding segments.

From the list of object files, every object module, referenced or not, is loaded resulting in Loader Symbol Table entries, a working storage section, and a binding section. Only referenced load modules are loaded.

Library segments may be shared by jobs. Programs using the same object file get separate object module segments built by the Loader.

The search order used by the Loader when resolving an external reference is as follows:

- o Loader Symbol Table
 - o Dictionary on each library in the order of the list. o Job Gate Table

2.2.2 TABLES

2.2.2.1 Program Control Block

The program control block (PCB) is an LNS structure used to define a program to the system. It has the following items: o Primary entry point - the name of the entry point at which to begin execution of the program. An alternate starting entry point can be specified in a task

control block. 46 o Binary object file list - the LNS name of a list of binary 47 object files, each of which containing one or more 48

NCR/CDC PRIVATE REV 30 APR 75

20 21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36 37

38

39 40

41

42

43

44

ADVANCED SYSTEM LABORATORY	CHP0604	2-5
ADVANGED STSTEN LABORATORT	CHF0804	75/05/21
IPLOS GDS - PROGRAM MANAGEMENT	~~~~~~	
2.0 PROGRAM EXECUTION		
2.2.2.1 Program Control Block		

- o Library list the LNS name of a list of libraries. Each library segment contains one or more load modules and dictionaries organized by entry point name that are used to locate procedures. All load modules in a library have the same segment attributes.
- o Size the initial working set size for the program. It is the number of page frames needed by any execution of the program when first brought into core by the Running Job Monitor.
- o Ring the ring of execution for the program. If specified, it must be within the execution bracket for all the files and segments specified in the PCB.
- Termination entry point an optional field specifying an 14 entry point name for a termination procedure. If 15 present, the termination procedure will be called by 16 the system during the orderly process of task 17 termination. Parameters will be passed indicating a 18 normal or abnormal termination. 19
- 2.2.2.2 Task Control Block

The task control block (TCB) is an LNS structure used to define the execution environment for a program. It has the following items:

- o PCB the LNS name of the program control block that defines the program to be executed by this task.
- o Entry point an alternate entry point name at which to 30 begin execution of the program. If specified, would 31 override the primary entry point as named in the 32 program control block. The alternate entry point 33 could change the definition of the program. 34
- o Size an alternate working set size for the program being 35 executed. If specified, would override the initial 36 working set size in the program control block. 37
- o Parameters the parameter block pointed to at entry.
- o Loader map options indicating the level of detail to be 39 generated for a loader map. 40
- o Abort options indicating the kind of dump required on an 41 abort.
- o Exit the type of exit (normal, abnormal) taken by this 43 task via the PM#EXIT request.
- o Code- an integer completion code specified on the PM#EXIT 45 request by this task. 46
- o Message a completion message up to 31 characters specified on the PM#EXIT request by this task.

NCR/CDC PRIVATE REV 30 APR 75

IDVANCED SYSTEM LABORATORY

0

1

2

3

4

5

6

7

8

9

10

11

12

13

20

21

22

23

24

25

26

27

28

29

38

42

44

47

48

2-6

PLOS GDS - PROGRAM MANAGEMENT

2.0 PROGRAM EXECUTION

2.2.2.2 Task Control Block

o EPCB - pointer to the Established Program Control Block for this task. Placed here by the Establisher.

CHP0 60 4

2.2.2.3 Established Program Control Block

The established program control block (EPCB) is a structure internal to Program Management and is used to define the loaded environment for a program. The EPCB can be the result of either a PM#EXECUTE request or a PM#ESTABLISH request and has the following items:

- o How established ~ indicates established by PM#EXECUTE or by PM#ESTABLISH.
- o TCB the LNS locator of the task control block specified on either request.
- o PCB the LNS locator of the program control block.
- o JCB the LNS locator of the job control block for the job in which the program is established. This field is used to obtain Job Gate Table and Job Stack Table entries for any further linking and stack allocation in this lob.

Ring - the ring in which the program is to be executed. 0

- Loader symool table pointer to the loader symbol table for this program.
- o Binary object file list the same list specified in the PCB but in a format more convenient for use by the loader.
- 0 Library list - the same list specified in the PCB but in a format more convenient for use by the loader.
- ٥ Thread - the EPCBs are threaded together on a job basis. The starting point is in the JCB.
- Keys the global and local key (not supported in V 1.0). 0
- o Event pointer to the event control block of the task completion event for the task as specified on the PM#EXECUTE request.
- o Control point the control point id for the task.
- o Dependencies task dependency threads for future use.
- o LNS search list pointer to the LNS search list for this established program.

2.2.2.4 Job_Gate_Table

.

The Job Gate Table (JGT) is a structure internal to Program Management and is used to register gated entry points on a job 47 basis. The JGT is searched by the Loader when resolving external 48

NCR/CDC PRIVATE REV 30 APR 75

2.2.2.4 Job Gate Table

references. An entry point is registered in the JGT during the loading of a module that possesses the gate attribute. All the entry points of such a module are registered as gates.

Gate is the mechanism used to satisfy the requirement of protecting one program. from another by allowing entry to the protected code at defined points. Not only does the Loader put a gated entry point in the JGT but also marks it in the binding section so the hardware can enforce the protection. The user cannot write a binding section.

2.2.2.5 Job Stack Table

The Job Stack Table (JST) is a structure internal to Program 16 Management and is used to allocate stacks when a control point is 17 created. It is a lob local array of integers indicating the 18 number of stacks to allocate on a ring-by-ring basis. When a job 19 is created, some minimum set of job and execution tables are 20 built. The JST is included in this job template specifying stack 21 allocation for Task Services. Using the PM#ESTABLISH request 22 will change the JST for the specified program establishment 23 ring. 24 25

2.2.3 TASK ESTABLISHMENT EXAMPLE

The purpose of the following example is to show structures visible to the user that make up the execution environment for his program. The example starts at the point execution is asked for via SCL, which in turn issues the request:

PM#EXECUTE (task, event, status)

task: the LNS descriptor of "USER_TCB" obtained by SCL via 37 the LNS#ENTRY request.

event: not used in this example.

status: request status returned to SCL.

Figure 2.2-1 shows the relationship of LNS structures 44 declared prior to issuing PM#EXECUTE. For these structures, the 45 diagram includes only those LNS fields necessary to the example. 46 47

o "USER TCB": local LNS name of the Task Control Block 48

NCR/CDC PRIVATE REV 30 APR 75

1

2

3

5

6

7

8

9

10 11

12

13

14

15

26

27 28

29

30

31

32

33

34

35

36

38

39

40 41

42

43

75/05/21

2-8

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

32

38 39

40

41

42

43

44

45

46

47

48

- IPLOS GUS PROGRAM MANAGEMENT
 - 2.0 PROGRAM EXECUTION

2.2.3 TASK ESTABLISHMENT EXAMPLE

specifying the program to execute and the parameters retrievable by that program.

CHP0 60 4

- o "USER_PCB": local LNS name of the Program Control Block defining the program via a list of object files and a list of library segments.
- o "USER OBJ LIST": local LNS name of list of object files generated by prior compilation. The object modules on these files will be converted to code sections in the object module segment.
- o "USER_LIB_LIST": local LNS name of the list of library segments to be used to search in the order listed for unresolved external references.
- o "OBJ_FILE_1" and "OBJ_FILE_2"; local LNS names of the File Control Blocks describing the object files to be loaded. The example has each file containing one module, A and B respectively.
- o "USER LIBRARY": local LNS name of a File Control Block describing the library file of the user. Only the referenced modules of this library will be loaded. The user can convert object files to library segments by using the library generator, OBLIGE.
- o "COBOL_RUN_TIME": global LNS name of a File Control Block 28 describing the library segment of COBOL run time 29 routines. This library segment is generated by the 30 installation and is shared by users. 31

Figure 2.2-2 shows the user segments created through program 33 loading for the PM#EXECUTE request. Note that working storage 34 sections and binding sections are created for every object module 35 but not every load module. Only the user stack segment is 36 shown. There would also be a stack segment allocated via the JST 37 for Task Services.

Figure 2.2-3 shows additions to user segments resulting from the request:

PM#LOAD (name, type, pointer, status)

name: name of an entry point in load module D on the user library file.

type: type of pointer to be used in a reference to the

ADVANCED SYSTEM LABORATORY	CHP0604	2-9
	0111 0004	75/05/21
IPLOS GDS - PROGRAM MANAGEMENT		

2.0	PROGRAM EXECUTION		
2.2.3	TASK ESTABLISHMENT EXA	1PLE	•
~~~~	_ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	,	

module.

pointer: the returned pointer after loading.

status: returned request status.

The load module D does not have any externals causing the loading of any other modules. Had it any, those load modules contained the matching entry points would have been loaded by PM#LOAD as well.



CHP0604

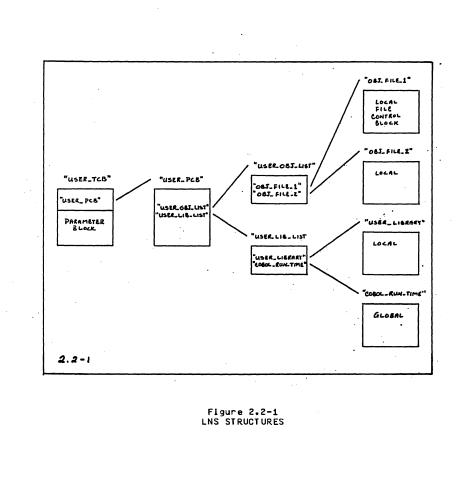
75/05/21

2-10

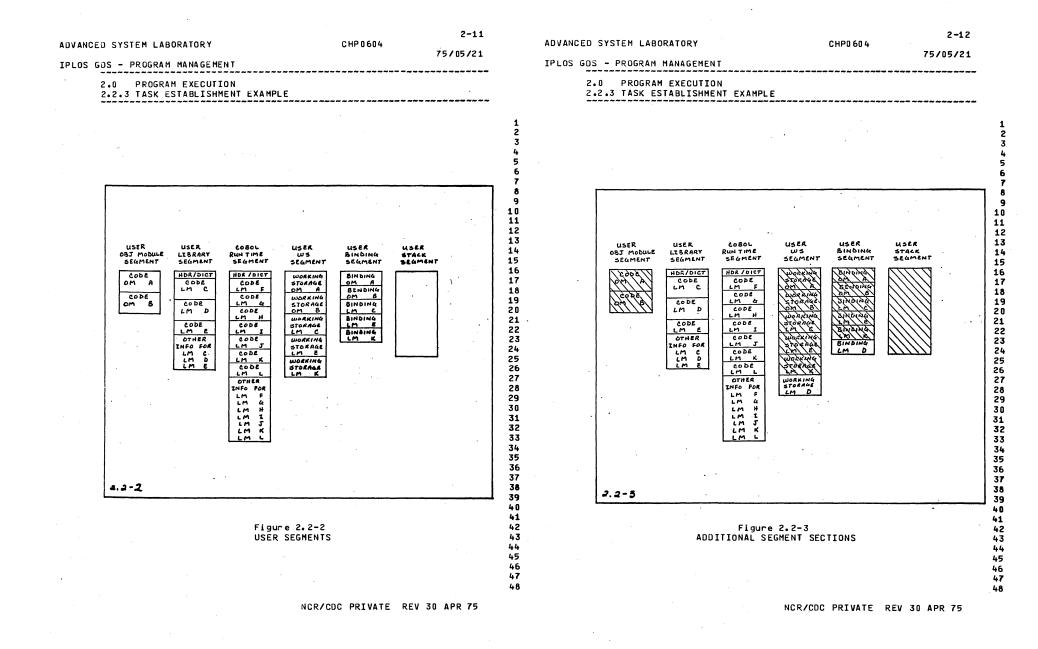
IPLOS GDS - PROGRAM MANAGEMENT

2.0 PROGRAM EXECUTION

2.2.3 TASK ESTABLISHMENT EXAMPLE



NCR/CDC PRIVATE REV 30 APR 75



ADVANCED SYSTEM LABORATORY IPLOS GDS - PROGRAM MANAGEMENT	CHP0 60 4	2-13 75/05/21	ADVANCED SYSTEM LABORATORY IPLOS GDS - PROGRAM MANAGEMENT	CHP
2.0 PROGRAM EXECUTION 2.2.4 SUBTASK ESTABLISHMENT	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~		2.0 PROGRAM EXECUTION 2.3 PROGRAM EXECUTION REQUESTS	~~~~~~~~~~
2.2.4 SUBTASK ESTABLISHMENT			1 2.3 <u>PROGRAM_EXECUTION_REQUESIS</u> 2 3	
2.2.4.1 <u>Subtask Control Block</u> The subtask control block declaration of which causes the a control point and the allocatio Stack Table. It has the followin o Control point - the cont Task Services in the System Task in the Sy has a return signal a response.	allocation and init n of stacks accordi g items: rol point id typica body of a signal be stem Job. That Sys	ialization of ing to the Job ally passed by ang sent to a item Task then when sending a	4       There are several levels         5       eventually exist for interfacing to         6       0       Command Language statemed         7       0       Control Language macros         8       0       Requests         9       0       Calls         10       11       Documentation for calls will detected         13       0       Request code         13       0       Refurence request status         14       0       Request block         15	program ents etail thr
			18Control Language macros may not20with the calls. There may be21Control Language. Likewise, there22macros not externalized through the23	some cal may be so Command
			24     The Control Language macros       25     follows: (To be supplied).       26     Request documentation is simply       27     Request documentation is simply       28     function performed and the paramet       29     Requests are one-to-one with cal       30     requests are follows:	y a pros ers supp
		•	3132PM#EXECUTE (task, event, status)33PM#EXIT (type, code, message)34PM#TERMINATE (task, status)35PM#SPAWN (entry, parameters, sub36PM#LOAD (name, type, pointer, status)37PM#RITRY (name, gate, segment, status)38PM#REINITIALIZE (name, status)39PM#ESTABLISH (task, status)40PM#DISESTABLISH (task, status)	btask, ev tatus)
			41 42 43 2.3.1 PM#EXECUTE 44 45 46 This request is used to load a 47 asynchronously execute that program 48	
	NCR/CDC PRIVATE RE			R/CDC PRI

75/05/21 -----1 2 3 everal levels of documentation that will interfacing to program execution: 5 6 7 8 9 10 for calls will detail three parameters in SWL: 11 12 13 14 15 ation will be provided as soon as request 16 17 18 ge macros may not necessarily be one-to-one 19 There may be some calls not visible in the 20 Ikewise, there may be some Control Language 21 ized through the Command Language. 22 23 anguage macros for program execution are as 24 25 26 ntation is simply a prose description of a 27 and the parameters supplied by the requestor. 28 o-one with calls. The program execution 29 30 31 32 33 34 parameters, subtask, event, status) 35 36 gate, segment, type, pointer, status) 37 38 39 40 41 42 43 44 45 s used to load a program and create a task to 46 47 48

2-14

CHP0604

2-15 ANDED SYSTEM LABORATORY CHP0604	ADVANCED SYSTEM LABORATORY CHP0604
ANCED SYSTEM LABORATORY CHPU604 75/05/21 OS GDS - PROGRAM MANAGEMENT	
2.0 PROGRAM EXECUTION 2.3.1 PM#EXECUTE	2.0 PROGRAM EXECUTION 2.3.4 PM#SPAWN
PM#EXECUTE (task, event, status)	1 2.3.4 PM#SPAWN
task: the LNS descriptor of a previously declared task control block used by the requestor to identify and control task execution. The task control block identifies the program control block of the program to be loaded and executed.	nd    4
event; optional parameter that is a pointer to an event control block to be associated with task completion. If specified, Program Management will cause the event wher	nt 9 entry: pointer to procedure at which to start asynchronous If 10 execution.
task completion is detected.	12 parameters: pointer to argument list for the procedure. 13
status: returned request status.	14subtask: the LNS descriptor of a previously declared subtask15control block, which resulted in allocations of a control16point and stacks.
2.3.2 PM#EXIT	17 18 status‡ returned request status.
This request is used to indicate task completion.	19 20 21 2.3.5 PM#LOAD
PM#EXIT (type, code, message)	22 23
type: indicates the type of exit being taken, normal or abnormal. The exit_type is put in the task control block by the PM#EXIT request processor.	or 24 This request is used to load a procedure not yet referenced ck - 25 in a program. 26 27 PM#LOAD (name, type, pointer, status)
code: a programmer defined integer put in the task control block by the PM#EXIT request processor.	
message: a programmer defined message up to 31 characters pu in the task control block by the PM#EXIT reques processor.	ut 31 st 32 type; the type of pointer to be returned. Can specify return 33 of a 48 bit pointer, a code base pointer, or a code 34 base-binding section pair. 35
2.3.3 PM#TERMINATE	36 pointer: returned pointer according to specified type 37 wanted. 38
This request is used by a task to terminate another task.	39 status: returned request status. 40
PM#TERMINATE (task, status)	41 42 2.3.6 PM#ENTRY
task: the LNS descriptor of the task control block of th task to be terminated. The TCB must be one used for previous PM#EXECUTE request.	he 43 a 44 45 This request is used to retrieve a pointer to be used in a 46 call to a specified entry point. The module containing the entry
status: returned request status.	47 point must have been previously loaded. The order of search for 48 the entry point is the same for loading (a Loader Symbol Table
NCR/CJC PRIVATE REV 30 APR 75	NCR/CDC PRIVATE REV 30 APR 75

2-17		
ADVANCED SYSTEM LABORATORY CHP0604 75/05/21		IDVANCED SYSTEM
IPLOS GDS - PROGRAM MANAGEMENT		PLOS GDS - PRO
2.0 PROGRAM EXECUTION 2.3.6 PM#ENTRY		2.0 PF 2.3.8 PM
and then the Job Gate Table).	1	tas
PM#ENTRY (name, gate, segment, type, pointer, status)	2 3	
name: entry point name.	45	
gate: optional parameter indicating search should be only on the Job Gate Table.	8	sta
segment: optional parameter indicating the segment which dictates the LST to start the search. Segment numbers in a job are unique per establishment of a program.		2.3.9 F
type: type of pointer to be returned. Can specify return of a 48 bit pointer, a code base pointer, or a code base-binding section pair.		The add PM#
pointer: returned pointer according to type.	17 18	tas
status: returned request status.	19 20	
	21 22	sta
2.3.7 PM#REINITIALIZE	23	
The purpose of this request is to provide COBOL an Operating System function necessary to satisfy their implementation of the ANSI standard CANCEL statement. This assumes the implementation of their CALL statement would use our PM#LOAD request. We do not currently know exactly what is required of the Operating System to satisfy any requirement imposed by a CANCEL implementation.	27 28 29	А 
PM#REINITIALIZE (name, status)	33	
name: entry point name.	34 35	
status: returned request status.	36 37 38 39	
2.3.8 PM#ESTABLISH	40 41 42	
This request is used to establish a program in the address space of a job. The primary purpose of the request is to establish subsystem services on a job basis.	43	
PM#ESTABLISH (task, status)	46 47 48	
NCR/COC PRIVATE REV 30 APR 75		

2-18 CHP0604 M LABORATORY 75/05/21 OGRAM MANAGEMENT -----ROGRAM EXECUTION #ESTABLISH ~~~~~~~~ k: the LNS descriptor of a previously declared task control block used by the requestor to identify and control the loading of a program. The task control block identifies the program control block of the program to be -4 loaded. atus: returned request status. PM#DISESTABLISH ۰. his request is used to remove an established program from iress space of the job. DISESTABLISH (task, status) sk: the LNS descriptor of the task control block describing the program that was established. atus: returned request status. 

NCR/CDC PRIVATE REV 20 MAY 75

3-1

75/05/21

ADVANCED SYSTEM LABORATORY

CHP0 60 4

IPLOS GDS - PROGRAM MANAGEMENT

3.0 LOGICAL NAME SPACE MANAGEMENT

3.0 LOGICAL NAME SPACE MANAGEMENT

This document is the GDS for the Logical Name Space manager for IPL/OS.

The functions described are the basic capabilities of the subsystem. As the OS requirements for LNS services become better defined, more sophisticated functions will be built using these basic capabilities. IPLOS GDS - PROGRAM MANAGEMENT

3.0 LOGICAL NAME SPACE MANAGEMENT 3.0.1 Design objectives

3.0.1 DESIGN OBJECTIVES

The design objectives of the Logical Name Space manager are as follows.

CHP0604

 To provide a generalized technique for the mapping of names to data.

• To provide a symbol table handler for System Command Language.

• To apply structuring methods to dynamic OS data compatible with SWL data representation for OS code and SCL for user manipulation• (i.e. records, arrays, etc.)

• To retain certain attributes of data to allow generic requests that may operate on several types of data or resources.

• To provide a degree of data protection and privacy by a hierarcical block structure of data segments while allowing the explicit sharing of data when required.

#### 3.1 SYSTEM DESCRIPTION

The logical name space (LNS) is composed of user and system supplied segments containing user and system defined entries. A list called the LNS segment list is maintained for each job known to the system. The LNS segment list contains the names of the segments which are to be searched for LNS entries and the order in which they are to be searched. When an LNS entry is sought each segment whose name appears in the LNS segment list is searched until the entry has been found or the list has been exhausted. The segment whose name appears in the last slot of the LNS segment list is called the most local segment and is searched first.

All internal LNS information uses relocatable addressing enabling a segment to be established at any virtual address while preserving previously defined information.

During job initiation the system allocates an LNS segment list and initializes it as follows.

LNS#GLOBAL

•

system global segment

NCR/CDC PRIVATE REV 20 MAY 75

NCR/CDC PRIVATE REV 20 MAY 75

3-2

75/05/21

		00
ADVANCED SYSTEM LABORATORY	CHP0604	
		75/05/21

# IPLOS GDS - PROGRAM MANAGEMENT

3.0	LOGICAL	NAME	SPACE	MANAGEMENT	
3.1	SYSTEM	DESCRI	PTION		
~~~		~~~~~			~~~~~~~~~~~~~~~~~

other segments

LNS#LOCAL most local segment

Each entry has an internal entry descriptor. These internal descriptors are managed in several chains per segment with a name hashing algorithm randomly assigning an entry to a chain. The entry search strategy includes a percolating of the internal descriptor chain which results in the chain being ordered by most recent use. Item chains are nandled in the same manner.

An entry and its internal descriptor form the primary node of a data structure through which the user can descend to any level.

ADVANCED SYSTEM LABORATORY

3-3

CHP0 60 4

75/05/21

3-4

IPLOS GUS - PROGRAM MANAGEMENT

3.0 LOGIC	AL NAME	SPACE	MANAGEMENT	
3.1.1 LNS	DESCRI	PTORS		

3.1.1 LNS DESCRIPTORS

Each entry or item in the LNS has an internal LNS descriptor associated with it which is NOT accessable to the user. The definition of this internal descriptor is as follows.

type_desc = RECORD desc_type: (entry, item), "type of descriptor" lock: BOOLEAN, "internal synchronization lock" chain: REL ~type_desc, "chain to next descriptor" name: STRING (31) OF CHAR, "name of entry or item" hash: 0..255, "hash value of name" excl: STRING (31) OF CHAR, "exclusive lock key" non_excl: 0..65565, "non-exclusive lock count" data_type: 0..max_type, "subscript to LNS#TYPE table" data_len: 0..max_len, "string or set length" data_time: 0..max_dim, "dimension of array variable" data: REL ~type_data, "location of data" ex_attr: SET OF 1..64, "extrinsic attributes" RECEND,

A complex type is described by an array of internal field descriptors. This array exists only once in the global segment regardless of the number of occurences of the complex type. The definition of the internal field descriptor is as follows.

type_field_desc = ARRAY [*] OF RECORD namet STRING (31) of CHAR, "name of field" hash: 0..255, "hash value of name" data_typet 0..max_type, "subscript to LNS#TYPE table" data_len: 0..max_len, "string or set length" data_dimt 0..max_dim, "dimension of array variable" data REL Type_data, "location of field in record" ex_attr: SET OF 1..64, "extrinsic attributes" RECEND,

Several of the LNS requests require or return a descriptor. This descriptor resides in the users memory and is fully accessable. The definition of this descriptor is as follows.

NCR/CDC PRIVATE REV 20 MAY 75

NCR/CDC PRIVATE REV 20 MAY 75

<pre>data_lent 0max_len data_dimt 0max_dim data_sizet 0max_si exclt 800LEAN, "exc non_exclt 800LEAN, datat ^type_data, " desct ^type_desc, "</pre>	pe, "subscript to LNS≢TY , "string or set length" , "dimension of array va ze, "size of data in cel	riable" Is" " riptor"	simple type or a comple type. The currently define UNKNOWN INTEGER REAL BOOLEAN	75/05/21 GEMENT
<pre>.1 LNS DESCRIPTORS type_user_desc = RECORD data_type: 0max_ty data_len: 0max_len data_dim: 0max_dim data_size: 0max_si excl: 800LEAN, "exc non_excl: B00LEAN, data: Type_data, " desc: Type_desc, " ex_attr: SET 0F 16</pre>	pe, "subscript to LNS#TY , "string or set length" , "dimension of array va ze, "size of data in cel lusive lock on" "non-exclusive lock(s) on location of data" location of internal desc	riable" Is" " riptor"	3.1.2 LNS DATA TYPES 3.1.2 LNS DATA TYPES LNS data types fall in simple type or a comple type. The currently define UNKNOWN INTEGER REAL BOOLEAN	nto two classes; simple and complex. ex type may be an element of a comple ed simple types are as follows. undefined type (undeclarable) integer variable real variable
data_type: 0max_ty data_len: 0max_len data_dim: 0max_len data_size: 0max_si excl: 800LEAN, "exc non_excl: B00LEAN, data: ~type_data, " desc: ~type_desc, " ex_attr: SET 0F 16	pe, "subscript to LNS#TY , "string or set length" , "dimension of array va ze, "size of data in cel lusive lock on" "non-exclusive lock(s) on location of data" location of internal desc	riable" Is" " riptor"	LNS data types fail in simple type or a comple type. The currently define UNKNOWN INTEGER REAL BOOLEAN	ex type may be an element of a comple ed simple types are as follows. undefined type (undeclarable) integer variable real variable
data_type: 0max_ty data_len: 0max_len data_dim: 0max_len data_size: 0max_si excl: 800LEAN, "exc non_excl: B00LEAN, data: ~type_data, " desc: ~type_desc, " ex_attr: SET 0F 16	pe, "subscript to LNS#TY , "string or set length" , "dimension of array va ze, "size of data in cel lusive lock on" "non-exclusive lock(s) on location of data" location of internal desc	riable" Is" " riptor"	LNS data types fail in simple type or a comple type. The currently define UNKNOWN INTEGER REAL BOOLEAN	ex type may be an element of a comple ed simple types are as follows. undefined type (undeclarable) integer variable real variable
data_dim: 0max_dim data_size: 0max_si excl: BOOLEAN, "exc non_excl: BOOLEAN, data: ~type_data, desc: ~type_desc, " ex_attr: SET OF 16	, "dimension of array va ze, "size of data in cel lusive lock on" "non-exclusive lock(s) on location of data" location of internal desc	riable" Is" " riptor"	simple type or a comple type. The currently define UNKNOWN INTEGER REAL BOOLEAN	ex type may be an element of a comple ed simple types are as follows. undefined type (undeclarable) integer variable real variable
ex_attr: SET OF 16	"non-exclusive lock(s) on location of data" location of Internal desc 4, "extrinsic attributes	" "	I NTEGER REAL BOOLEAN	integer varlable real variable
ex_attr: SET OF 16	location of data" location of internal desc 4, "extrinsic attributes	riptor" "	I NTEGER REAL BOOLEAN	integer varlable real variable
ex_attr: SET OF 16	4, "extrinsic attributes		REAL BOOLEAN	real variable
			STRING	character string variable
			SET	set variable
			POINTER	pointer variable
			CELL	cell variable
				LNS allas name
			UHAIN	LNS item chain
	· · · · ·		The following addition	nal subsets of INTEGER will be include
	х.		when their IPL/SWL represen	ntations are defined.
			SUBRANCE	i)
				(X, Y, Z, D, LNS)
			CASENCE	, / , . , . ,
				representation attributes are no the LNS design.
			PACKED	
			O RAIMED	
			The currently defined	complex types are as follows.
			SCL#TOKEN	SCL token
·			SCL# OPERATOR	SCL operator
			SCL#FUNCTION	SCL function
			SCL#COMMAND	SCL command
			SCL#MACRO	SCL macro
				• •
	NCR/CDC PRIVATE RE	V 20 MAY 75		NCR/COC PRIVATE REV 20 MAY 75
		NCR/CDC PRIVATE RE	15 16 17 18 19 20 21 23 24 25 26 27 28 29 30 31 31 32 33 34 34 35 36 37 38 39 40 40 41 42 43 44 43 44 43 44 44 35 46 47 48 NCR/CDC PRIVATE REV 20 MAY 75	16 ALIAS 17 CHAIN 18 The following addition 19 when their IPL/SML represent 21 SUBRANGE 22 SUBRANGE 23 ORDINAL 24 The following SML 25 The following SML 26 Supported at this stage of 27 PACKED 28 PACKED 29 CRAMMED 30 The currently defined 31 The currently defined 32 SCL#OPERATOR 35 SCL#OPERATOR 36 SCL#CONNAND 37 SCL#MACRO 38 39 40 41 42 43 44 45 45 46

ADVANCED SYSTEM LABORATORY CHP0604	ADVANCED SYSTEM LABORATORY CHP0604
75/05/21 IPLOS GDS - PROGRAM MANAGEMENT	IPLOS GDS - PROGRAM MANAGEMENT
3.0 LOGICAL NAME SPACE MANAGEMENT 3.1.3 LNS STRUCTURES	3.0 LOGICAL NAME SPACE MANAGEMENT 3.1.3.1 General Examples
3.1.3 LNS STRUCTURES	1 The following structure is built for scalar string 2 variables. 3
By the use of cataloged internal descriptors of complex types and chained items, arbitrarily complex structures may be assembled.	4 eeeeeee 5 ddddddd 6 \$ 7 \$.>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
	9 Example: 10 ins#declare name,string 11
Examples of the commands to build the structures shown are included using Command Language syntax for clarity. The following symbols are used in the diagrams of these structures.	12 The following structure is built for string arrays. 13 14 eeeeeee 15 ddddddd
e entry name f field name i item name d LNS internal descriptor c chain linkage	16 : 17 :.>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
The following structure is built for scalar numeric variables.	22 Example: 23 Ins#declare name,string,32,4 24 25
eeeeeee ddddddd	26 27 28
\$•>xxxxxxx	29 30
Example: Ins#declare name,integer	31 32 33
The following structure is built for numeric arrays. eeeeeeee ddddddd	34 35 36 37
₿。>xxxxxxx xxxxxxxx xxxxxxx xxxxxxx xxxxxx	38 39 40 41 42
Example: Ins#declare name,real,dim=4	43 44 45 46 47 48
NCR/CDC PRIVATE REV 20 MAY 75	NCR/CDC PRIVATE REV 20 MAY 75

ANCED SYSTEM LABOR	RATORY	СН	P0 60 4	3-9	ADVANCED SY	STEM LABORATORY
OS GDS - PROGRAM	IANAGEMEN T			75/05/21	IPLOS GUS -	PROGRAM MANAGEMENT
3.0 LOGICAL N/ 3.1.3.1 Genera	AME SPACE MANA al Examples	GE ME N T				LOGICAL NAME SPACE MAN 3.1 General Examples
The fol	lowing structu	re is built for	complex types	•	1 2 are	As illustrated below permitted.
eeeeeee					3	• • • • • •
dddddd					4 5	eeeeeee ddddddd
	•	<pre>dddddddd</pre>			5	1
		<ddddddd< td=""><td></td><td></td><td>7</td><td>* • >xxxxxx</td></ddddddd<>			7	* • >xxxxxx
		< ffffffff			8	*****
	:	tddddddd			9	•••cccccc
	:				10	\$ xxxxxx
		>1111111			11	1
		••• ccccccc•••			12 13	2
	1	1 1			14	>111111
	*.>xxxxxxxx	*. >XXXXXXXXX *	> x x x x x x x x x		15	cccccc
					16	• • • ddd dd dd
Example:		-			17	1
	ord record_nam				18	
	ld record_name	,field_a,integer			19 20	• • •
		,field_c,chain			21	1
Ins#dec	are structure	,type=record_name	•		22	·>xxxxxxx
Ins#inse	ert structure.	field_c,item_a			23	*****
		field_c,item_b			24	******
Ins#Inse	ert structure.	field_c,item_c			25 26	******
					27	
					28	
	+				29 70 Eve	!-!
			ь. Х		30 Exa 31	mple: Ins#record subrecord
					32	ins#field subrecord_
					33	ins#field subrecord_
					34	ins#record suprecord
					35	Ins#field subrecord_
					36	Ins#field suprecord_
					37 38	Ins#record record_na Ins#field record_nam
					39	Ins#field record_name
					40	Ins#field record_nam
				4	41	Ins#field record_nam
					42	Ins#declare name,typ
					+3	Ins#insert name.fiel
					+4 45	Ins#insert name.fiel Ins#insert name.fiel
					45	ins#insert name.fiel
					47	
					+8	

NAGEMENT w, all combinations of items and fields ••••>ffffffff ...ddddddd xx<.1 ffffffff xx<....ddddddd cc<... ffffffff xx<. :.dddddddd tfffffff t....ddddddd ii ••>iiiiiiii ••>iiiiiiii cc...: ccccccc..: ccccccc dd ...ddddddd ... dddddddd 1 1 *.>XXXXXXXX *.>CCCCCCC... . •••••>ffffffff 11111111..: ...ddddddd ccccccc xx<.1 ffffffff dd dddd d d... xx<....ddddddd... 1 xx<. 1 : xx x x x x x x < . : xx 1 1 ffffffff.1 t :.dddddddd tfffffff too ddddddddd d_b,2 _b,field_a,real _b,field_b,dim=2 1_a,2 _a,field_a,integer _a,field_b,type=subrecord_b ame,4 me,field_a,integer me,field_b,real me,field_c,chain me,field_d,string,8 pe=record_name d_c,item_a,type=subrecord_a d_c,item_b d_c,item_c,type=chain d_c.item_c,item_a,real

NCR/CDC PRIVATE, REV 20 MAY 75

5

6

7

8

9

10

11

12 13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28 29 30

31

32 33

34 35

36 37

38

39

40

41

42

43 44

45

46 47 48

3-10

75/05/21

CHP0604

VANCEJ SYSTEM LABORATORY CHPO	3-11	ADVANCED SYSTEM LABORATORY	CHP0604	3-12
LOS GDS - PROGRAM MANAGEMENT	75/05/21	IPLOS GDS - PROGRAM MANAGEMENT		75/05/21
				~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
3.0 LOGICAL NAME SPACE MANAGEMENT 3.1.3.2 SCL#TOKEN Example		3.0 LOGICAL NAME SPACE MANAGEMENT 3.1.3.2 SCL#TOKEN Example		
3.1.3.2 <u>SCL#IOKEN_Example</u>		1 Ins#field scl#string,lhi 2 Ins#field scl#string,rni 3 Ins#field scl#string,bufi	•string,255	
SWL Definition		4 5 Instrecord scittoken,5		
ТҮРЕ		6 Ins#field scl#token,typ 7 Ins#field scl#token,desc	ins#desc	
Ins#desc = RECORD		8 Ins#field_scl#token,iv		
data_type: 0max_type,		g ins#field_scl#token;rv;r	al	
data_len: 0max_len,	1	0 Ins#field scl#token,sv,s	:l#string	
data_dim: 0max_dim,	1	1		
data_size: 0max_size,	1	2		
excl: BOOLEAN,	1	3		
non_excl: BOOLEAN,	1	4		
data: ^type_data,	1	5		
desc: ^type_desc,	1	6		
ex_attr: SET OF 164,	1	7		
RECEND,	1			
	1			•
scl#string = RECORD,		0		
lhi: 1256,		1		•
rhi <b>:</b> 0255,	2			
buff: STRING (255) OF CHAR,	2			
RECEND,		4		
		5		
scl#token = RECORD		6		
typ: INTEGER,	2			
desc: ins#desc,	2			
iv: INTEGER,		9		
rv: REAL,	3	-		
sv: scl#string,	3			
RECEND;	3			
		3		
LNS_Definition		4 5		
	3			
Ins#record Ins#desc,9,(declare,insert),1		o 7		
Ins#field Ins#desc,data_type		8		
Ins#field Ins#desc;data_Ipe		9	:	
Ins#field Ins#desc,data_dim	4	-		
Ins#field Ins#desc,data_size		1		
Ins#field Ins#desc,excl,boolean		2		
<pre>Ins#field Ins#desc,non_excl,boolean</pre>		3		
Ins#field Ins#desc,data,pointer	4			
Ins#field Ins#desc,desc,pointer	4			
Ins#field Ins#desc,ex_attr,set,64	. 4			
Ins#record scl#string,3,(declare,insert)	•Ins#103 4			
	, <u></u>	-		
	VATE REV 20 MAY 75		NCR/COC PRIVATE	DEN 20 MAY 75

		3-13
ADVANCED SYSTEM LABORATORY	CHP0604	
		75/05/21
IPLOS GDS - PROGRAM MANAGEMENT		

7 B LOCTOR NAME SPACE MANAGEMENT	
3.0 LOGICAL NAME SPACE MANAGEMENT	
3.1.4 TYPE CONTROLLED "OWN CODE" PROCEDURES	
3.1.4 ITPE CUNIRULLED OWN CODE PROCEDORES	•

3.1.4 TYPE CONTROLLED "OWN CODE" PROCEDURES

Provision has been made for "own code" procedures external to the LNS system code to be conditionally called by request and data type. This feature is intended to allow other components of the operating system to be advised of certain LNS operations on data with which they are concerned. These procedures will be passed the LNS request code, the request parameters and the LNS internal descriptor of the entry or item to be acted upon. A 10 status must be returned by the procedure. If this status is not 11 normal, the LNS request will be aborted and the status returned 12 to the user. All parameters and status records will follow .13 normal system conventions. 14

These procedures are defined at the time the internal field 16 descriptor is catalogued by the LNS#RECORD request. The dynamic 17 loader is called with the named procedure to supply a value for a 18 procedure pointer variable (^PROC) in the LNS#TYPE table. This 19 procedure is called by the specified LNS requests whenever a 20 variable or a field of a variable of the specified type is 21 referenced. The trap procedure is invoked prior to any action by 22 the LNS request with the exception of LNS#DECLARE and 23 LNS#INSERT. These two requests call the procedure after 24 completing their respective functions. A trap procedure may 25 issue LNS requests. However, recursion and interlock problems 26 are possible if the logic of the trap is defective. 27

Two examples of possible uses of these procedures are to initialize variables when declared or inserted or to monitor changes via get and put by the user to tables currently in use by the system. A third example would be the implicit declaration of associated variables or insertion of items into a chain field.

A set of procedures named in the form LNS#<status code> are 35 supplied in the LNS library. These procedures function as own 36 code trap routines and return the status record indicated by 37 their name. For example LNS#103 returns a status of "invalid 38 type" and may be used to prevent declaration of a complex type 39 intended only for use as a field of another complex type and 40 never as a variable on its own. This is illustrated in the 41 SCL#TOKEN example.

ADVANCED SYSTEM LABORATORY

3-14

1

2

3

q

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

75/05/21

IPLOS GDS - PROGRAM MANAGEMENT

3.1.5 EXTRINSIC ATTRIBUTES
3.0 LOGICAL NAME SPACE MANAGEMENI
3

#### 3.1.5 EXTRINSIC ATTRIBUTES

1

2

3

5

6

7

8

9

15

28

29

30

31

32

33

34

42

43

44

45

46

47

48

In addition to the normal LNS intrinsic attributes, the LNS system will retain 64 user defined extrinsic attributes for any element. These attributes have no meaning to the LNS system but may be assigned and queried by the user.

Attributes 33 to 64 are reserved for operating system use (i.e. SCL decoding attributes) while attributes  $\tilde{1}$  to 32 may be manipulated by the end user (i.e. problem program).

Currently reserved attributes are:

#### 33..40

#### system command language

CHP0604

#### 3.2 LNS REQUESTS

The following requests are available for the manipulation of LNS.

NCR/COC PRIVATE REV 20 MAY 75

PLOS 63 - PROCEDIM MANAGEMENT       75/05/21         3.1 USIGAL MAR SPACE MANAGEMENT       3.2 (J LUSANTIACH         J.2.1 LUSANTIACH       3.2 (J LUSANTIACH         The purpose of the LNSANTIACH request is to add a new segment is to the LNSANTIACH request is to add a new segment is to asguent occursely is segment.       3.2 (J LUSANTIACH         Segment is to segment is to add a new segment is to add a new segment is to response of the LNSANTIACH request is to resolve a segment list.       3.2 (J LUSANTIACH         Segment is descent parameter boscifies a string containing the name of a segment currently known to the jeb (Lee in a part of the veriable is to add as centre) to the string containing the is segment all be initialized as eably.       3.2 (J LUSANTIACH cagament is to be cliced. USANTIACH request is to resolve a segment is to be part of the veriable is to be cliced. The string containing the is segment all be initialized as eably.         Old The old parameter concilies a veriable. If the isolut record is to be placed. The string containing the string	VANCED SYSTEM LABORATORY	CHP0604	3-15	ADVANCED SYSTEM LABORATORY	CHP0604	3-16
3.2.1 LNSAATTACH 3.2.1 LNSAATTACH 3.2.2 LNSADETACH The europse of the USAATTACH request is to add a new segment to the LNS segment, list as the lost local segment. LNSAATTACH (segment, list as the lost local segment. Segment to the segment parameter specifies a string containing the name of a segment currently known to the job (i.e. napped in). Old The old parameter specifies a boolean variable. If the value of the variable is true, the LNS data currently in the segment mill be initialized as eachy. Status The status parameter specifies a variable is the status record is to be placed. The status code returned are described under "error conditions". 333 335 337 337 337 337 348 349 341 345 345 345 345 345 345 345 345	PLOS GUS - PROGRAM MANAGEMENT		75/05/21	IPLOS GDS - PROGRAM MANAGEMENT		75/05/21
The purpose of the LNSARTACH request is to add a new segment to the LNSARTACH (segment, list as the most local segment. LNSARTACH (segment, old, status) segment The segment parameter specifies a string containing is segment in the segment parameter specifies a bolean variable. If the variable is frue, the LNS data currently is faise, the segment will be accessable. If the variable is frue, the LNS data currently is faise, the segment will be placed. The status codes a returned are described under "error conditions".		MENT	· · · · · · · · · · · · · · · · · · ·			
The purpose of the LNSMITACH request is to add a new segment to the LNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the UNSMITACH request is to remove a segme from the unstant remove a segme from the UNSMITACH request is to remove a segme from the unstant remove a segme from the segment full is the remove a segme from the unstant remove a segme from the segment full is the statement remove a segme from the segment full is the statement remove a segme from the segment full is the statement remove a segme from the segment full is the statement remove a segme from the segment full is the statement remove a	3.2.1 LNS#ATTACH					
LUSARTACH (segment, old, status) segmenti The segment parameter specifies a string containing the name of a segment currently known to the job (i.e. mapped in). oldi The old parameter specifies a bolean variable. If the segment will be accessable. If the variable is the status parameter specifies a variable into which the status parameter specifies	The purpose of the L segment to the LNS segment Li	LNS#ATTACH request is ist as the most local s	to add a new a egment•	The purpose of the LNS#DETACH from the LNS segment list.	request is to re	move a segmer
segment i The segment parameter specifies a string containing 10 the name of a segment currently known to the job (i.e. mapped in). old The old parameter specifies a boolean variable. If the value of the variable is true, the LiS data currently to alse, of the variable is true, the LiS data currently to alse, the segment will be initialized as empty. status The status parameter specifies a variable into which the status parameter specifies a variable into which the status record is to be placed. The status cord is	LNS#ATTACH (segment, old	d, status)	1	LNS#DETACH (segment, status)		
<pre>value of the variable is true, the LVS data currently 15 status The status parameter specifies a variable in 16 which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status parameter specifies a variable into which 19 the status param</pre>	the name of a segm mapped in).	nent currently known to	ing containing 1 the job (i.e. 1 1 1	segment: The segment parameter the name of the segmen this parameter (indicated	nt to be detached I by a blank stri	<ul> <li>Omission d</li> </ul>
status The status parameter specifies a variable into which is the status record is to be placed. The status codes returned are described under "error conditions". 23 24 25 26 27 28 30 31 31 32 33 34 35 35 36 37 38 39 40 41 41 42 43	value of the vari in the segment will	iable is true, the LNS (   be accessable. If the	data currently 19 e variable is 10 empty. 11	status: The status parameter s the status record is to b returned are described ur	e placed. The	status code
23 24 25 26 27 28 29 30 31 32 33 34 35 35 35 35 37 39 41 41 41 41 41 41 41 41 41 41	the status record i	is to be placed. The	ple into which 1 status codes 20 lions". 22	3 1 1		
26         27         28         29         30         31         32         33         34         35         36         37         38         39         40         41         42         43         44         45         46         47			23	5		
28 29 30 31 32 33 34 35 36 37 38 39 40 41 41 41 41 42 43 43 43 43 43 44 43 44 45 46			- 26			
30 31 32 33 33 34 35 36 37 38 39 40 41 42 43 43 44 45 45 46 47 48			21			
32 33 34 35 36 37 38 39 40 41 42 42 43 44 43 44 43 44 43 44 43 44 43 44 45 46 47 48			30			
34 35 36 37 39 40 41 41 42 43 43 44 45 46 47 48						
36 37 38 39 40 41 42 43 43 44 45 46 45 46 47 48						
38 39 40 41 42 43 44 45 46 46 47 48	· · ·					
39 40 41 42 43 43 44 45 46 46 47 48						
41 42 43 44 45 46 47 48			× 39	· ·		
43 44 45 46 47 48			41			
45 46 47 48			43			
47 48			45	i		
			47			
NCR/CDC PRIVATE REV 20 MAY 75 NCR/CDC PRIVATE REV 20 MAY 7!		· · · · · · · · · · · · · · · · · · ·				
		NCR/CDC PRIVATE RE	V 20 MAY 75	NC	R/CDC PRIVATE R	EV 20 MAY 75
			•			

OS GDS - PROGRAM MANAGEMENT 75/05/2	IPLOS GOS - PROGRAM	M MANAGEMENT		75/05/21
	~~~~~~~~~~		~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	
3.0 LOGICAL NAME SPACE MANAGEMENT 3.2.3 LNS#DECLARE	3.0 LOGICAL 3.2.4 LNS#RI	NAME SPACE MANAGEMENT Emove 		
3.2.3 LNS #DECLARE	1 3.2.4 LNS#	REMOVE		·
The purpose of the LNS#DECLARE request is to declare a entry in the LNS.	3 4 The 1 5 from the Lt	purpose of the LNS#REMOVE NS•	request is to re	nove an entry
LNS#DECLARE (segment, entry, type, length, dim, status)		EMOVE (segment, entry, sta	tus)	
segment: The segment parameter specifies a string containing the name of the segment in which the entry is to in declared. Omission of the segment parameter (indicate by a blank string) will cause the entry to be declare in the most local segment.	11 12 13 14	nt: The segment parameter the name of the segment wh entry. Omission of the se a blank string) will c appears in the LNS segment	ich is to be sea gment parameter ause each segmen	rched for the (indicated by nt whose name
entry: The entry parameter specifies a string containing t name of the entry being declared.	17 1	: The entry parameter spec name of the entry which is		ontaining th
type: The type parameter specifies a string containing th type of the entry being declared. Omission of the typ parameter (indicated by a blank string) will cause a entry of type INTEGER to be declared. The valid LN types are those described under "data types" or ar complex type previously defined by LNS#RECORD ar LNS#FIELD.	20 21 22 23 24 25	s: The status parameter sp the status record is to be returned are described und	placed. The	status code
length: The length parameter is only meaningful whe declaring string or set variables. For strings th parameter specifies an integer containing the number of bytes to be allocated for the string. For se variables the integer contains the number of element in the set. Omission of the length paramete (indicated by a D) will cause a default of 32 to b assumed.	26 27 28 29 30 31 32 33 34			
dim: The dim parameter specifies an integer containing th dimension of the entry being declared. Omission of th dim parameter (indicated by a 0) will cause a defaul of 1 to be assumed.	35 36 37 38 39			
status: The status parameter specifies a variable into whic the status record is to be placed. The status code returned are described under "error conditions".	40 41 42 43 44 45 46			
	47 48			
NCR/CDC PRIVATE REV 20 MAY 75		NCR	COC PRIVATE RE	/ 20 MAY 75

/

ADVANCED SYSTEM LABORATORY CHP0604	3-19	ADVANCED SYSTEM LABORATORY CHP0604
IPLOS GDS - PROGRAM MANAGEMENT 75/	/05/21	IPLOS GDS - PROGRAM MANAGEMENT 75/05/21
3.0 LOGICAL NAME SPACE MANAGEMENT 3.2.5 LNS#ENTRY		3.0 LOGICAL NAME SPACE MANAGEMENT 3.2.6 LNS≉NEXT
3.2.5 LNS#ENTRY	1	
The purpose of the LNS#ENTRY request is to ge descriptor of an LNS entry given the name of the entry.	5	of a field or item given the descriptor of the enclosing entry, item or field.
<pre>LNS#ENTRY (segment, entry, subscr, desc, status) segment: The segment parameter specifies a string cont the name of the segment which is to be searched i entry. Omission of the segment parameter (indica a blank string) will cause each segment whose appears in the LNS segment list to be searched. entry: The entry parameter specifies a string containin name of the entry whose descriptor is being sough subscript to be used when the entry is an Omission of the subscr parameter (indicated be will cause a descriptor of the entire array returned. desci The desc parameter specifies a record into wh descriptor is to be returned. status: The status parameter specifies a variable into the status record is to be placed. The status returned are described under "error conditions".</pre>	7 8 9 100 the 11 11 11 11 11 11 11 11 11 11 11 11 11	<pre>LNS#NEXT (input_desc, name, subscr, output_desc, status) input_desc: The input_desc parameter specifies the name of a record containing a descriptor of the enclosing entry, field or item. name: The name parameter specifies a string containing the name of the field or item whose descriptor is being sought. subscr: The subscr parameter specifies an integer containing the subscript to be used when the field or item is an array. Omission of the subscr parameter (indicated by a 0) will cause a descriptor of the entire array to be returned. output_desc: The output_desc parameter specifies a record</pre>
NCR/CDC PRIVATE REV 20 MAY	48	NCR/CDC PRIVATE REV 20 MAY 75

• .

VANCED SYSTEM LABORATORY CHP0604				YSTEM LABORATORY	CHP0604	3-22
OS GDS - PROGRAM MANAGEMENT		75/05/21	IPLOS GDS -	- PROGRAM MANAGEMENT		75/05/21
3.0 LOGICAL NAME SPACE MANAGEMENT 3.2.7 LNS#SLICE	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~			LOGICAL NAME SPACE MANAGEN .8 LNS#GROW	ient	
3.2.7 LNS#SLICE			1 3.3	2.8 LNS#GROW		
The purpose of the LNS#SL1 descriptor of an element of an arr array.	CE request is ay given the desc	to get the riptor of the		The purpose of the LNS≇G an LNS entry or item.	ROW request is to grow t	the dimensior
LNS#SLICE (input_desc, subscr,		• . •	7 8	LNS#GROW (desc, incr, st	tatus)	
input_desc: The input_desc p containing the descriptor	arameter specifi	1 es a record 1 be sliced. 1	1 2	desc: The desc parame descriptor of the e be grown.	eter specifies a record entry or item whose dimen	
subscr: The subscr parameter s the subscript of the desi	red element.	. 1	4 5 6	grown. Omission o	the dimension of the en of the incr parameter (in	ry is to b dicated by
output_desc: The output_desc p into which the descrip placed.	arameter specifi tor of the ele	es a record 1 ment is to be 1 1 2	8	status: The status param	ault of 1 to be assumed meter specifies a variab 1 is to be placed. The	e into whic
status: The status parameter s the status record is t returned are described un	o be placed. The	status codes 2 ions"• 2	2 3	returned are descri	ped under "error condit:	ons".
		2 2 2	5			
		2 2 2	3			
		3 3 3	-			
		3 3 3 3	5 			
		3 3 3				
		3 4 4)			
		44 44 44	la de la companya de Se de la companya de l			
• • •		44 44 44				•
	R/CDC PRIVATE REV	4				

D SYSTEM LABORATORY CI	3-23 HP0604	ADVANCED SYSTEM LABORATORY CHP0 60 4
DS - PROGRAM MANAGEMENT	75/05/21	75/05/21
3.0 LOGICAL NAME SPACE MANAGEMENT 3.2.9 LNS#LOCK		3.0 LOGICAL NAME SPACE MANAGEMENT 3.2.10 LNS#UNLOCK
3.2.9 LNS#LOCK		1 3.2.10 LNS#UNLOCK 2 3
The purpose of the LNS#LOCK request or item. The locking operation has no e except other LNS#LOCKs, LNS#UNLOCK, LNS#RI	effect on other requests	4 The purpose of the LNS#UNLOCK request is to unlock an LNS 5 entry or item. 6
LNS#LOCK (desc, excl, key, status)		7 8 LNS#UNLOCK (desc, key, status) 9
desc: The desc parameter specifies a descriptor of the entry or item	a record containing a	10desc: The desc parameter specifies a record containing a11descriptor of the entry or item to be unlocked.12
excl: The excl parameter specifies the value of the variable is tru will be exclusive. If the valu be non-exclusive.	a boolean variable. If ue, accass to the entry ue is false, access will	13key:The key parameter specifies a string containing the14string returned by the LNS#LOCK request when the entry15was locked for exclusive access. If the entry is being16unlocked from non-exclusive access the key parameter is17ignored.
key: The key parameter specifies a st in which a unique name will be r requested was exclusive. If nor requested, the contents are unci	tring of 31 characters returned when the access n-exclusive access was	1819status! The status parameter specifies a variable into which20the status record is to be placed. The status codes21returned are described under "error conditions".22
status: The status parameter specific the status record is to be place returned are described under "en	es a variable into which ed. The status codes rror conditions".	23 24 25 26
		27 28 29
		30 31
		32 33
		34
		36 37
		38 39
		40 41
		42
		43 44
		45 46
		47 48
	S	NCR/CDC PRIVATE REV 20 MAY 75

ADVANCED SYSTEM LABORATORY CHP0604 75/05/21	3-26 NDVANCED SYSTEM LABORATORY CHP0604 75/05/21 IPLOS GDS - PROGRAM MANAGEMENT
IPLOS GDS - PROGRAM MANAGEMENT 3.0 LOGICAL NAME SPACE MANAGEMENT 3.2.11 LNS#INSERT	IPLOS GDS - PROGRAM MANAGEMENT 3.0 LOGICAL NAME SPACE MANAGEMENT 3.2.12 LNS#DELETE
3.2.11 LNS#INSERT	1 3.2.12 LNS#DELETE 2
The purpose of the LNS#INSERT request is to insert a new item into a chain.	3 4 The purpose of the LNS#DELETE request is to delete an item 5 from a chain. 6
LNS#INSERT (desc, item, type, length, dim, status)	7 8 LNS#DELETE (desc, item, status)
desc: The desc parameter specifies a record containing a descriptor of the entry, field or item within which the item is to be allocated.	9 10 desc: The desc parameter specifies a record containing a 11 descriptor of the entry, field or item which is to be 12 searched for the item.
item: The item parameter specifies a string containing the name of the item to be allocated.	13 14 item: The item parameter specifies a string containing the 15 name of the item to be deleted.
type: The type parameter specifies a string containing the type of the item being inserted. Omission of the type parameter (indicated by a blank string) will cause an item of type INTEGER to be inserted. The valid LNS types are those described under "data types" or any complex type previously defined by LNS#RECORD and LNS#FIELD.	16 status: The status parameter specifies a variable into which 17 the status record is to be placed. The status codes 18 returned are described under "error conditions": 20 21 21 NOTE: The trap to an "own code" procedure by the LNS#DELETE 23 request is determined by the data type of the item being 24 deleted.
length: The length parameter is only meaningful when inserting string or set items. For strings the parameter specifies an integer containing the number of bytes to be allocated for the string. For set items the integer contains the number of elements in the set. Omission of the length parameter (indicated by a 0) will cause a default of 32 to be assumed.	25 26 27 28 29 30 31
dim: The dim parameter specifies an integer containing the dimension of the item being inserted. Omission of the dim parameter (indicated by a D) will cause a default of 1 to be assumed.	32 33 34 35 36
status: The status parameter specifies a variable into which the status record is to be placed. The status codes returned are described under "error conditions".	37 38 39 40 41
NOTE: The trap to an "own code" procedure by the LNS#INSERT request is determined by the data type of the item being inserted.	42 43 44 45 46 47
NCR/CUC PRIVATE REV 20 MAY 75	48 NCR/CDC PRIVATE REV 20 MAY 75

ADVANCED SYSTEM LABORATORY	CHP0604	3-27	3-28 ADVANCED SYSTEM LABORATORY CHP0604
	CHP0604	75/05/21	75/05/21
IPLOS GDS - PROGRAM MANAGEMENT			IPLOS GDS - PROGRAM MANAGEMENT
3.0 LOGICAL NAME SPACE MANAGEMENT 3.2.13 LNS#GET			3.0 LOGICAL NAME SPACE MANAGEMENT 3.2.14 LNS#PUT
3.2.13 LNS#GET			1 3.2.14 LNS#PUT 2
The purpose of the LNS#GET request the LNS.	tistoget a	value from	3 4 The purpose of the LNS#PUT request is to put a value into 5 the LNS. 6
LNS#GET (desc, buffer, status)			7 8 LNS#PUT (desc, buffer, status) 9
desc: The desc parameter spec descriptor of the entry, fie being sought.			10 desc: The desc parameter specifies a record containing a 11 descriptor of the entry, field or item whose value is 12 to be updated.
buffer: The buffer parameter spe the value is to be placed.	ecifies a buffer	• into which	15 the new value.
status: The status parameter spec the status record is to be returned are described under	placed. The s	status codes	18 the status record is to be placed. The status codes 19 returned are described under "error conditions".
			20 21 22
			23 24 25 26 27
			28 29 30 31
			32 33 34
			35 36 37
			38 39 40
			41 42 43 44
			45 46 47 48
NCR/CL	C PRIVATE REV	20 MAY 75	48 NCR/CJC PRIVATE REV 20 MAY 75

.

з

R

75/05/21

ADVANCED SYSTEM LABORATORY

3.0 LOGICAL NAME SPACE MANAGEMENT 3.2.15 LNS#SETXA

3.2.15 LNS#SETXA

The purpose of the LNS#SETXA request is to set the extrinsic attributes of an entry or item. Permission to alter attributes 33..64 is verified by the OS#CHECK procedure.

LNS#SETXA (desc, attr, status)

desc: The desc parameter specifies a record containing a descriptor of the entry or item whose extrinsic attributes are to be set.

CHP0604

- attr: The attr parameter specifies a set of 1...64 containing the attributes to be changed. This set will be "xored" to the current set of attributes resulting in the symetric difference of the two sets. In other words, the presence of any attribute in this parameter causes the attribute to be "toggled" in the LNS internal descriptor.
- status: The status parameter specifies a variable into which the status record is to be placed. The status codes returned are described under "error conditions".

[PLOS GDS - PROGRAM MANAGEMENT

3.0 LOGICAL NAME SPACE MANAGEMENT 3.3 PRIVILEGED REQUESTS

3.3 PRIVILEGED_REQUESTS

The following requests are subject to restrictions such as Operating System only or SE# OP use. When any of these requests are issued permission is verified by the OS#CHECK procedure.

CHP0604

NCR/CDC PRIVATE REV 20 MAY 75

NCR/COC PRIVATE REV 20 MAY 75

3-30

INCED SYSTEM LABORATORY CHP0604 DS GDS - PROGRAM MANAGEMENT	3-31 75/05/21		NCED SYSTEM Is GDS - PROC	LABORATORY	IT	CHP0604	757	3-32 05/21
3.0 LOGICAL NAME SPACE MANAGEMENT 3.3.1 LNS#RECORD			3.0 LOGIC 3.3.1 LNS	CAL NAME SPACE S#RECORD	MANAGEMENT			
3.3.1 LNS#RECORD		1 2			record is to e described	be placed. The under "error cond	e status ditions".	codes
The purpose of the LNS#RECORD request is to complex type to the system. The type defininglobal.		3 4 5 6 7				· .		
LNS#RECORD (record, fields, traps, procedure, s	status)	8 .9 10						
record: The record parameter specifies a strin the name of the complex type to be defined		11 12			•			
fields: The fields parameter specifies an integ the maximum number of fields to exist in type.		13 14 15 16 17						
traps: The traps parameter specifies an or requests for which an "own code" procedur invoked for this type. If this paramet (indicated by an empty set) no traps will positional significance of each request as follows.	re is to be ter is omitted occur. The	18 19 20 21 22 23						
LNS#DECLARE		24 25 26 27 28 29 30 31 32 33 32 33 34 35						
L NS#PU T	· .	36 37 38						
procedure: The procedure parameter specific containing the name of the "own code" pr invoked as indicated by the traps par procedure named must reside in a libr known to the job. An error in this pa result in a status being returned fr rather than from LNS. If this parameter (indicated by a blank string) no traps will	rocedure to be ameter. The ary currently arameter will rom the loader is omitted ll occur.	39 40 41 42 43 44 45 46 46 47						
status: The status parameter specifies a variab	le into which	48						

VANCED SYSTEM LABORATORY	C HP0 60 4	3-33	ADVANCED SYSTEM LABORATORY	CHP0604	3-34 75/05/21
LOS GDS - PROGRAM MANAGEMENT		75/05/21	IPLOS GUS - PROGRAM MANAGEMENT		
3.0 LOGICAL NAME SPACE MANAGEMENT 3.3.2 LNS#FIELD			3.0 LOGICAL NAME SPACE MANAGE 3.3.3 LNS#SEGLOCK	EMENT	~~~~~~~~~~
3.3.2 LNS#FIELD			1 3.3.3 LNS#SEGLOCK 2		
The purpose of the LNS#FIELD re a previously defined complex type.	equest is to define		6 non-exclusive lock on a seg 6 request from being performed	LNS#SEGLOCK request is t ment in order to prevent ar d.	o perform a LNS#DETACH
LNS#FIELD (record, field, type,	, len, dim, attr, st	atus)	7 8 9 LNS#SEGLOCK (segment,	status)	
record: The record parameter sp the name of the complex t be a member.		ield is to 1 1	1 segment: The segment p 2 the name of the Li	arameter specifies a string NS segment to be locked.	containing
field: The field parameter spec name of the field to be de the name of the first curr complex type.	efined. This name w	ill become 1	4 status‡ The status par 5 the status reco 6 returned are desc 7	ameter specifies a variable rd is to be placed. The s ribed under "error conditic	status codes
type: The type parameter spec type of the field to be de parameter (indicated by field of type INTEGER to b types are those describ complex type previously LNS#FIELD.	efined. Omission of a blank string) wi be defined. The bed under "data typ	raining the 1 the type 2 the cause a 2 valid LNS 2 pes" or any 2	9 0 1 2 3 4 5		
length: The length paramete defining string or set parameter specifies an int bytes to be allocated for the integer contains th set. Omission of the leng 0) will cause a default of	fields. For str reger containing the the string. For s ne number of eleme gth parameter (indic	ngful when 2 rings the 2 number of 2 set fields 3 ents in the 3 sated by a 3 3	7 8 9 0 1 2		. •
dim: The dim parameter specif dimension of the field bei dim parameter (indicated of 1 to be assumed.	ing defined. Omissi	aining the 3 on of the 3	5 6 7 8		
attr: The attr parameter specif the extrinsic attributes field. Note that the LNS# on field descriptors.	s to be associate	containing 4 ed with the 4 not be used 4 4	0 1 2 3		
status: The status parameter sp the status record is to be returned are described unc	e placed. The sta	atus codes 4	5 6 7		
NCE	R/CDC PRIVATE REV 2	0 MAY 75		NCR/CDC PRIVATE REV	20 MAY 75

AD VANCED	SYSTEM LABORATORY	CHP0604	3-35 75/05/21		ADVANCED SYSTEM LABORATORY
IPLOS GUS	S - PROGRAM MANAGEMENT				IPLOS GDS - PROGRAM MANAGEME
	0 LOGICAL NAME SPACE MANAG 3.4 LNS#Segunlock				3.0 LOGICAL NAME SPAC 3.4.1 DEFINITION OF C
3	3.3.4 LNS#SEGUNLOCK			1 2	3.4.1 DEFINITION OF
2	The purpose of the LNS segment, allowing an LNS#D6			3 4 5	0 LN 000 norma
	<u>.</u>			6 7	Parameter errors
	LNS#SEGUNLOCK (segment	t, status)		8	8 LN 101 inval 8 LN 102 inval
	segment: The segment p	parameter specifies a	string containing	10	8 LN 102 INVa 8 LN 103 inva
		NS segment to be unl		11	8 LN 104 inva
	status: The status par	ometer encelfier o v	onichie iste which	12 13	8 LN 105 inva 8 LN 106 inva
		ord is to be placed.		13	8 LN 108 INVA 8 LN 108 INVA
		criped under "error c		15	8 LN 109 inva
				16	8 LN 10A inva
-	3.4 ERROR_CONDITIONS			17 18	Access errors
				19	Access en 013
				20	8 LN 201 deni
_	Error conditions are			21	8 LN 202 segme
	returned by each LNS re bassed to the system messag			22 / 23	8 LN 203 segm 8 LN 204 entr
	logging.	ge generator for furt		24	8 LN 205 entr
	55 5			25	8 LN 206 field
				26	8 LN 207 field
		Ý		27	8 LN 208 item
				28 29	8 LN 209 item
				30	Functional errors
				31	
				32	8 LN 301 entr
				33 34	8 LN 302 entr 8 LN 303 segm
				35	8 LN 303 Segmi 8 LN 304 elem
				36	8 LN 305 elem
				37	8 LN 306 elem
				38	8 LN 307 segm
				39 40	Internal errors
				41	Three har en or 3
				42	C LN 901 no m
				43	C LN 902 no m
				44 45	C LN 903 maxi C LN 904 maxi
				45	C LN 904 maxi
				47	E LN EEE feat
				48	F LN FFF disa
		NCR/COC PRIVAT	E REV 20 MAY 75		

1 2 3

Ŀ

5 6

7 8

37 38 39

75/05/21

OS GDS - PROGRAM MANAGEMENT

3.0 LOGICAL NAME	SPACE MANAGEMENT	
3.4.1 DEFINITION	05 00055	
2.4.1 DELINTITON	01 00023	

3.4.1 DEFINITION OF CODES

0 LN 000 normal completion

8	LN	101	invalid	segment name
8	LN	102	invalid	element name
8	LN	103	invalid	
8	LN	104	invalid	length
8	LN	105	invalid	dimension
8	LN	106	invalid	increment
8	LN	108	invalid	key
8	LN	109	invalid	subscript
8	LN	10A	invalid	descriptor

8	LN 201	. denied access
8	LN 202	segment exists
8	LN 203	segment does not exist
8	LN 204	entry exists
8	LN 205	entry does not exist
8	LN 206	field exists
8	LN 207	' field does not exist
8	LN 208	item exists
8	LN 209) item does not exist

8 LN 301	entry already locked		
8 LN 302	entry not locked		
8 LN 303	segment locked by system		
8 LN 304	element not a chain		
8 LN 305	element not a structure		
8 LN 306	element too large		
8 LN 307	segment not locked		
	•		
Internal errors			

nternal errors		40
		41
C LN 901 n	o memory for LNS internal descriptor	42
	no memory for data	43
	naximum number of fields exceeded	44
	aximum number of segments exceeded	45
	aximum number of types exceeded	46
	eature not yet supported	47
F LN FFF d		48

NCR/CDC PRIVATE REV 20 MAY 75

ANCED SYSTEM LABORATORY CHP0604 DS GDS - PROGRAM MANAGEMENT		ADVANCED SYSTEM LABORATORY 75/05/21 IPLOS GDS - PROGRAM MANAGEMENT		CHP0604	75/05/21
3.0 LOGICAL NAME SPACE MANA 3.4.2 ERROR CODES BY REQUES		· ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	3.0 LOGICAL NAME SPACE MANAGEM 3.4.2 ERROR CODES BY REQUEST	ENT	
3.4.2 ERROR CODES BY REQUE	ST	1		P I S I R I F I S I S I	
	D \$ R \$ E \$ N \$ S \$ G \$ L \$			U : E : E : I : E : E : T : T : C : E : G : G : : X : O : L : L : U :	
# T # T #	E \$ E \$ N \$ E \$ L \$ R \$ O \$ C \$ M \$ T \$ X \$ I \$ O \$ C \$ _ \$ O \$ R \$ T \$ C \$ W \$ K \$		1 R 1 T 1 1 1 T 1 E 1 1 1 1 E 1 1	1 A 1 R 1 D 1 O 1 N 1 1 D 1 B C 1 L 1 1 1 1 1 K 1 O 1	
	AIVIYI IEI I	C # 9 K # 10			
••••••••••••••••••••••••••••••••••••••		11 11 12 13	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5		
* 8 LN 101 * X * * * 8 LN 102 * * *	<pre></pre>	X 14 15 15 16	\$ 8 LN 101 \$ \$ 8 \$ 8 LN 102 \$ X \$ X \$ \$ \$ 8 LN 103 \$ X \$ \$ \$ 8 LN 103 \$ X \$ \$	1 1 1 1 X 1 X 1 1 1 X 1 X 1 X 1 1 1 X 1 X	
\$ 8 LN 103 \$ \$ \$ \$ 8 LN 104 \$ \$ \$ \$ 8 LN 105 \$ \$		17 18 19	: 8 LN 104 ; X ; ; ; ; 8 LN 105 ; X ; ; ; ; 8 LN 105 ; X ; ; ; ; ; 8 LN 106 ; ; ; ; ;		
* 8 LN 106 * * * * 8 LN 108 * * * * 8 LN 109 * * *		* 20 X * 21 * 22	: 8 LN 108 : : : : : 8 LN 109 : : : : :		
* 8 LN 10A * * * * 8 LN 201 * * * * 8 LN 202 * X * *		X 8 23 8 24	8 8 LN 10A 8 X 8 X 8 X 8 8 8 LN 201 8 8 8 8 8 8 LN 202 8 8 8 8 8 LN 202 8 8 8		
* 8 LN 203 * * * * 8 LN 204 * * * 8 LN 204 * *		25 26 27	8 LN 203 8 8 8 8 8 8 LN 204 8 8 8 8 8 LN 205 8 8 8 8 8 LN 205 8 8 8	1 1 1 1 X 1 X 1 1 1 X 1 1 1 1 1 1 1 1 1	
* 8 LN 206 * * * * 8 LN 207 * * *		28 29 20 20	18 LN 2061111 8 LN 20711111 8 LN 2081X111		
* 8 LN 208 * * * * 8 LN 209 * * * * 8 LN 301 * *		1 31 1 32 1 33	\$ 8 LN 209 \$ \$ X \$ \$ \$ 8 LN 301 \$ \$ \$ \$ \$ 8 LN 302 \$ \$ \$ \$ 8 LN 302 \$ \$ \$		
\$8 LN 302 \$. \$. \$ \$8 LN 303 \$ \$X \$ \$8 LN 304 \$ \$		X 8 34 8 35 8 36	\$ 8 LN 303 \$ \$ \$ \$ \$ 8 LN 304 \$ X \$ X \$ \$		
* 8 LN 305 * 2 * * 8 LN 306 * * * * 8 LN 307 * * *		t 37 t 38	8 8 LN 305 8 8 8 8 8 8 LN 306 8 X 8 8 X 8 8 8 LN 307 8 8 8 8 8 LN 307 8 8 8		
C LN 901 1 1 1 C LN 902 1 1		8 39 8 40 8 41	\$ C LN 901 \$ X \$ \$ \$ \$ C LN 902 \$ X \$ \$ \$ C LN 902 \$ X \$ \$ \$ C LN 903 \$ \$ \$	1 1 X 1 8 8 8 1 1 X 1 2 1 8 1 8 8 X 8 8	
IC LN 903 I I I IC LN 904 IX I I IC LN 905 I I	i i i i i i i i i i i i i i i i i i i i	1 42 1 43 1 44	CLN 904 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2		
* E LN EEE * * * * F LN FFF * X * X * X *****		\$ 45 X \$ 46	5 F LN FFF 5 X 5 X 5 X 5 X 5 X 5 X 5 X 5 X 5 X	X I X I X I X I X I X I	
		48			

NCR/CDC PRIVATE REV 30 APR 75

2

7

1.

E

6

7

8

a

4.0

11

12

13

14

15

16

17

18

19

20

21

22

23

25

27

29

31

32

41

44

75/05/21

TPLOS GOS - PROGRAM MANAGEMENT

4.0 PROGRAM COMMUNICATIONS

4.0 PROGRAM_COMMUNICATIONS

4.1 EVENTS

Events are system supported facilities which permit synchronization and interrupt control for asynchronous activities within a Job. An event is represented by an event control block in storage and several system requests to manipulate the control plock. An event control block may be either an LNS variable or a structure in the job data base (internal static, stack, etc.).

The event control block contains the following information: o Condition state (caused, cleared) o Action state (enabled, disabled)

o Action (attached procedure, waited)

The condition state indicates the current condition of the 24 event, caused or cleared. The action state, enabled or disabled. directs the system to either immediately effect the specified 26 action or delay the action. The action can be the invoking of an attached procedure, or it can be continuing the execution of an 28 asynchronous activity in the Job that has been waiting for the causation of this event. The action may also be a combination of 3.0 both for one or several control points in a Job.

Regardless of the action state being either "enabled" or 33 "disabled", the system performs the specified action only when 34 the condition state of the event changes from 'cleared' to 35 "caused". There are two requests that do this change. 36 PM#CAUSE_EVENT and PM#CAUSE_CLEAR_EVENT. The PM#CAUSE_EVENT 37 request sets the condition state to caused and leaves it that 38 way. The PM#CAUSE_CLEAR_EVENT request is used for pulsing. If 30 the condition state of an event is 'caused' when either of these 4١ requests is issued, the system will not perform the specified action. The PM#CLEAR EVENT request sets the condition state of 42 an event to "cleared". 43

An interrupt procedure can be attached to an event by using 45 the PM#ATTACH PROCEDURE request. When an event to which an 46 interrupt procedure has been attached does occur, the result will 47 be the serial invocation of the attached procedure using the same 48

NCR/CDC PRIVATE REV 30 APR 75

ADVANCED SYSTEM LABORATORY

75/05/21

IPLOS	GDS - PROGRAM		MANAGEMENT
	4.0	PROGRAM	COMMUNICATIONS

4.1 EVENTS

point as the requestor of the PM#ATTACH_PROCEDURE control request.

CHP0604

If the action state of the event to which the procedure is attached is "disabled", the invocation of the procedure will be delayed until the event is enabled. Event occurrence processing for a particular event remains disabled until enabled by the PM#ENABLE EVENT request.

The PM#DISABLE_EVENT request prevents the invocation of any and all procedures attached to a particular event. A procedure can be attached to more than one event. More than one procedure can be attached to one event.

NCR/CDC PRIVATE REV 30 APR 75

ED SYSTEM LABORATORY CHP0604 75/05/2	ADVANCED SYSTEM LABORATORY CHP0604 75/05/21
GUS - PROGRAM MANAGEMENT	IPLOS GOS - PROGRAM MANAGEMENT
4.0 PROGRAM COMMUNICATIONS 4.1.1 EVENT REQUESTS	4.0 PROGRAM COMMUNICATIONS 4.1.1.2 PM#CAUSE_EVENT
4.1.1 EVENT REQUESTS	1 event: pointer to event control block. 2
The event requests provided by Program Management are follows:	3 status: returned request status. 4 5
PM#ATIACH_PROCEDURE (procedure, event, status)	6 4.1.1.3 <u>PM#CAUSE_CLEAR_EVENT</u> 7 8
<pre>PM#CAUSE_EVENT (event, status) PM#CAUSE_CLEAR_EVENT (event, status) PM#CLEAR_EVENT (event, status) PM#DETACH_PROCEDURE (procedure, event, status) PM#DISABLE_EVENT (event1,, eventM, status) PM#ENABLE_EVENT (event1,, eventM, status) PM#STATUS_EVENT (event, condition_state, action_stat waited, attached_interrupt_procedure, status) PM#WAIT_EVENT (event1,, eventM, position, status) PM#WAIT_CLEAR_EVENT (event1,, eventM, position, status)</pre>	9 This requests performs the action, if any, as specified 10 the event control block and returns to the requestor with th 11 event in the cleared state. If the event is in the caused stat 12 already when this request is made, the system does not perfor 13 any specified action and informs the requestor via the return
4.1.1.1 PM#ATTACH_PROCEDURE	19 event: pointer to event control block. 20
	21 status: returned request status. 22
This request establishes an association of an interru procedure with an event.	23 24 4.1.1.4 <u>PM#CLEAR_EVENT</u> 25
PM#ATTACH_PROCEDURE (procedure, event, status)	26
procedure: pointer to the procedure to be invoked when t event occurs.	27 This request sets the condition state of an event to 28 cleared. 29
event: pointer to event control block.	30 PM#CLEAR_EVENT (event, status) 31
status: returned request status.	32 event: pointer to an event control block. 33
	34 status⊧ returned request status. 35
4.1.1.2 <u>PM#CAUSE_EVENI</u>	36 37 4.1.1.5 <u>PM#DETACH_PROCEDURE</u> 38
This request sets the specified event to caused. If t event is in the cleared state when this request is made, t system performs the action, if any, as specified in the eve control plock. If the event is in the caused state already wh this request is made, the system does not perform any specifi action and informs the requestor via the returned reque status. Performing the action includes the execution of a	393940This request removes the association of an interrup41procedure with an event. The requestor must be the same as the42PM#ATTACH requestor.434444PM#DETACH_PROCEDURE (procedure, event, status)45
attached procedures. PM#CAUSE_EVENT (event, status)	46 procedure: pointer to procedure to no longer be associate 47 with the specified event control block. 48
NCR/CDC PRIVATE REV 30 APR 75	40 NCR/CDC PRIVATE REV 30 APR 75
NORYODU FRIVATE REV JU AFR 79	NUCKIOU FRIVATE REV JU AFR / J

ADVANCED SYSTEM LABORATORY CHP0604	4-5	ADVANCED SYSTEM LABORATORY CHP0604
IPLOS GDS - PROGRAM MANAGEMENT	75/05/21	IPLOS GDS - PROGRAM MANAGEMENT
4.0 PROGRAM COMMUNICATIONS 4.1.1.5 PM#DETACH_PROCEDURE		4.0 PROGRAM COMMUNICATIONS 4.1.1.8 PM#STATUS_EVENT
event: pointer to event control block.	1	in the Job waiting for this event to be caused.
status: returned request status. 4.1.1.6 <u>PM#DISABLE_EVENT</u>	3 4 5 6 7	attached_interrupt_procedure: returned indication if there are any interrupt procedures that are attached to this event. 7
This request disables event occurrence proces event or events. It sets the action state of a sy to disabled.) 4.1.1.9 <u>PM#WAIT_EVENT</u>
PM#DISABLE_EVENT (event1,, eventM, status)	13	3
event: pointer to an event control block.	14 15	one or all of a specified number of events has occurred.
status: returned request status.	16 17	PM#WAIT_EVENT (event1,, eventM, position, status)
4.1.1.7 PM#ENABLE_EVENT	18 19 20 21 22	event: pointer to an event control block.) position: if specified, one event occurrence will satisify
This request enables event occurrence proces event or events. It sets the action of a speci enabled.	ssing for an 23 ified event to 24 25	specified, all the events must occur to satisfy the wait.
<pre>PM#ENABLE_EVENT (event1,, eventM, status)</pre>	26 27	
event: pointer to an event control block. status: returned request status.	28 29 30 31	not remain suspended waiting for something that will not occur. Elaosed default time limit will be reflected in the returned request status.
4.1.1.8 PM#STATUS_EVENT	32 33 34 35	While a control point is suspended waiting on an event, other events can occur. These are saved until the wait is satisfied. Then they are processed in the order of their
This request returns the status of an event.	36 37 38	wait. Processing event occurrences includes invoking any
PM#STATUS_EVENT (event, condition_state, waited, attached_interrupt_procedure, status	action_state, 39 s) 40) The PM#WAIT_EVENT request processor does not alter the
event; pointer to event control block.	41 42	
condition_state: returned state indicating cleared.	45	in which case all are suspended until the condition state of the specified event is caused.
action_state: returned state indicating disabled.	46 enabled or 47 48	If a task is suspended waiting on an event and there occurs
NCR/CDC PRIVATE REV	/ 30 APR 75	NCR/CDC PRIVATE REV 30 APR 75

1234567

36 37 38

39 40 41

42

43

48

			4-7
ADV	ANCED SYSTEM LABORATORY	CHP06	75/05/21
IPL	OS GUS - PROGRAM MANAGEM		
	4.0 PROGRAM COMMUNI	CATIONS	

4.1.1.9 PM#WAIT EVENT

attached, the system will allow the control point to execute the interrupt procedure and then be suspended again.

4.1.1.10 PM#WAIT_CLEAR_EVENT

This request suspends the execution of a control point until one or all of a specified number of events has occurred. It is the same as the PM#WAIT_EVENT request except that it returns to the requestor with the condition state of the wait satisfying event or events as cleared.

PM#WAIT_CLEAR_EVENT (event1, ..., eventM, position, status)

event: pointer to event control block.

position: if specified, one event occurrence will satisfy the wait and its position (1-M) will be returned. If not specified, all the events must occur to satisfy the wait.

status: returned request status.

The system will default a time limit so a control point will not remain suspended waiting for something that will not occur. Elapsed default time limit will be reflected in the returned request status.

If a control point is suspended waiting on an event and there occurs another event to which an inner-ring interrupt subprogram has been attached, the system will allow the control point to execute that interrupt subprogram and then be suspended 33 again.

36 While a control point is suspended waiting on an event, other events can occur. These are saved until the wait is satisfied. Then they are processed in the order of their occurrence including the event or events that satisfied the wait. Processing event occurrences includes invoking any attached interrupt procedures.

More than one control point may wait on an event, in which case all are suspended until the condition state specified event is caused.

NCR/CDC PRIVATE REV 30 APR 75

ADVANCED SYSTEM LABORATORY

75/05/21 IPLOS GDS - PROGRAM MANAGEMENT ------4.0 PROGRAM COMMUNICATIONS

CHP0 60 4

4.2 SIGNALS

1 2

3

q

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

34

35

37

38

39

40

41

42

43

44

45

46

47

48

4.2 ____SIGNALS

Signals are short messages that are used for inter-job communications usually in the form of requests and responses. For example, system code in a User Job can send a signal to the System Job to request some specific service. The body of the signal would contain the request. It may also contain the identification of an associated event control block for an event to be caused when a response is received.

A signal may be associated with a queue via the PM#SELECT_SIGNAL request. In this case Program Management would:

- 1) put the signal on a queue using the PM#ENQUEUE request
- 2) cause the queue-associated event, if any, as noted in the queue control block.

The signal can be removed from the queue by using the PM#DEQUEUE request.

When a signal is received by the destination control point, control first goes to a general signal handler and then is routed to signal-own-code based on the type of signal. For example, say I/O is a type of signal. Then every I/O signal received by a job could be processed by an I/O signal module to do whatever is particular for an I/O signal.

The types of signals and the information contained in a signal are detailed in Chapter 9 of OSGDS.

4.2.1 SIGNAL SELECTION

The Signal Selection List (SSL) is a structure internal to Program Management and is used to register signal selections on a control point basis. The PM#SELECT_SIGNAL request associates a signal with a queue by creating an entry in the SSL. The Task Monitor Signal Handler uses the SSL to queue signals. The PM#DESELECT_SIGNAL request removes an entry from the SSL.

46

47

48

NCR/CDC PRIVATE REV 30 APR 75

4-8

ADVANCED SYSTEM LABORATORY	4-9 CHP0604	ADVANCED SYSTEM LABORATORY CHP0 60 4
IPLOS GDS - PROGRAM MANAGEMENT	75/05/21	75/05/21 IPLOS GDS - PROGRAM MANAGEMENT
4.0 PROGRAM COMMUNICATIONS 4.2.2 SIGNAL REQUESTS		4.0 PRUGRAM COMMUNICATIONS 4.2.2.3 PM#DESELECT_SIGNAL
4.2.2 SIGNAL REQUESTS	•	1 4.2.2.3 <u>PM#DESELECT_SIGNAL</u> 2
follows: PM#SEND_SIGNAL (signal, sta		3 4 This request breaks the association of a signal with a 5 queue. Further receptions of the specified signal will not 6 result in those signals being items on the previously specified 7 queue.
PM#SELECT_SIGNAL (name, que PM#DESELECT_SIGNAL (name, s PM#STATUS_SIGNAL (signal, c	status)	8 9 PM#DESELECT_SIGNAL (name, status) 10
PM#DISABLE_SIGNALS (status) PM#ENABLE_SIGNALS (status)		11 name: the type and id of the signal as specified in a 12 previous PM#SELECT_SIGNAL request. 13
4.2.2.1 <u>PM#SEND_SIGNAL</u>		14 status: returned request status. 15
This request sends a signa	al from one job to another job.	16 17 4.2.2.4 <u>PM#STATUS_SIGNAL</u> 18
PM#SEND_SIGNAL (signal, st	tatus)	19 20 This request provides a way to determine if a particular 21 signal has arrived. This is meaningful for the case of more than
signal: pointer to the sign	nal to be sent.	21 signal has arrived. This is meaningful for the case of more than 22 one signal associated with a queue. It can be determined if 23 anything is on the queue by using the PM#STATUS_QUEUE request.
status; returned request st	tatus.	24 It can be determined if a particular signal is on a queue by 25 using the PM#STATUS_SIGNAL request.
4.2.2.2 PM#SELECI_SIGNAL		26 27 PM#STATUS_SIGNAL (signal, queue, status) 28
	a signal with a queue. More than ith one queue, but not multiple	29 name: type and id of a signal as previously specified in a 30 PM#SELECT_SIGNAL request. 31
queues for a signal. PM#SELECT_SIGNAL (name, qu	ueue. status)	32 queue: returned pointer to queue control block, if any, as 33 specified on a previous PM#SELECT_SIGNAL request. 34
— · ·	and id in the header of the signal	35 status‡ returned request status. 36 37
	e control block to be used by queue the signal so it will not be	38 4.2.2.5 <u>PM#DISABLE_SIGNALS</u> 39 40 41 This request is used to prevent loss of control due to
status: returned request s	tatus.	42 Interruption for signal processing for the requesting control 43 point. This request does not prevent hardware interruptions. 44
• • • •		45 PM#DISABLE_SIGNALS (status) 46
•		47 status: returned request status 48
	NCR/CDC PRIVATE REV 30 APR 75	NCR/CDC PRIVATE REV 30 APR 75

ANCED SYSTEM LABORATORY	CHP0 60 4	4-11	ADVANCED SYSTEM LABORATORY CH	4-12 1P0604 75/05/21
OS GDS - PROGRAM MANAGEMENT		/5/05/21	IPLOS GUS - PROGRAM MANAGEMENT	
4.0 PROGRAM COMMUNICATIONS 4.2.2.6 PM#ENABLE_SIGNALS	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~		4.0 PROGRAM COMMUNICATIONS 4.3 QUEUES	
4.2.2.6 <u>PM#ENABLE_SIGNALS</u>			4.3 <u>QUEUES</u>	
This request is used to ena previous PM#DISABLE_SIGNALS request.		ing after a	The queuing mechanism provided by designed to allow the sending, storing arbitrary data structures between asynchr a Job. The queuing facility will be used	g, and retrieving of conous activities within
PM#ENABLE_SIGNALS (status)			3 Signal mechanism to pass standard signal control points. An event may be associated	
status: returned request status.			an enqueue request on the queue would event. It is the responsibility of the op in the pointer to an associated ECB.	effect causation of the
4.2.2.7 <u>PM#IDENTITY</u>			3	
``````````````````````````````````````			A queue is defined by a queue contro address space of the Job. It can be	
This request is used to obtain requestor. The execution identity the task control block, the program	may be the control	ol point id,	somewhere in the job data base (stack) The format of a queue control block is sho	internal static, etc.)
control block.	control brocky a	, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	B TYPE	
PM#IDENTITY (to be supplied)			D QUEUE_CONTROL_BLOCK = RECORD NUMBER_QUEUED:SEMAPHORE; 2 ASSOCIATED_EVENT: ~EVENT_CONTROL_E	
			CHAIN_START: ^QUEUE_ITEM, CHAIN_START: ^QUEUE_ITEM, CHAIN_END: ^QUEUE_ITEM, STORAGE_METHOD: (To Be Supplied)	SLUCK,
			6 RECEND;	
			7 TYPE 3 QUEUE_ITEM = RECORD	
			9 QUEUE_THREAD: ~QUEUE_ITEM, DATA_LOCATION: ~SEQUENCE, RECEND.	
			2	
			3 The fields in the QUEUE_CONTROL_BLOCK described below: 5	and QUEUE_ITEM are
			6 NUMBER_QUEUED: This semaphore shoul 7 of zero. It indicates the number 8 the queue. Adding an item to	of items currently or
			9 SIGNAL_SEMAPHORE on this semaphore 0 from the queue will do a PH	and getting an item
			1 semaphore. 2	,
			ASSOCIATED_EVENT: If the pointer is references an event control block the caused state whenever there ar 6 and the cleared state whenever the	which will be placed in re items on the queue
			7	•••
:			8 CHAIN_START: Pointer to the first ite	em on the queue.

•

ANC	ED SYSTEM LABORATORY			СНЕ	20 60 A	4			4-	13	ADVA
	GOS - PROGRAM MANAGEMENT								/05/		IPLOS
	4.0 PROGRAM COMMUNICATIONS	~~~~	~~~~	·~~~			~~~~	~~~~			
	4.3 QUEUES										
	CHAIN_END: Pointer to the	e las	† 1†	em or	n the	e qu	eue.			1 2	
	STURAGE_METHOD: will inc to be acquired for new					ay w	here	sto	rage	is 3 4 5	
	QUEUE_THREAD: Thread of i	tems	on	the d	queue	e.				6	
	DATA_LOCATION: Pointer 1	to t	he	data	rep	pres	ente	d b	y		
	QUEUE_ITEM.									9 10	
	The actions performed				requ	uest	pro	cess	ors	are 11	
	described in the following dec	:1510	n tai	blei						12 13	
	+									14 + 15	
	OPERATION	1	ENQ	JEUE		1	DEQ	UEUE		1 16	
	I NUMBER OF ITEMS ON QUEUE	1	0	<b>i &gt;</b> (	)	1	0	: >	0	1 18	
	ASSOCIATED EVENT										
	+		+	+	+	+	+		+		
	1	1	1			i	i		i i	23	
	1	1				1	i	1 X 1	1	1 25	
	Suspend requesting process				1 	1 X 1	1 X 1	1	-	1 26 1 27	
	PM#CAUSE_EVENT	I X				1	1	1	-	1 28 1 29	
	PM#CLEAR_EVENT	1	i	1	l	•		i * '	1	1 30	
	l +	 -+	l +	 +	 	1 +	i +	: +	1 +	1 31 + 32	
										33 34	
		¥	if.	init	ial	valu	e is	one	•	35	
										36 37	
										38 39	
										40	
										41 42	
										42	
										44 45	
										45	
										47 48	
										+0	

D SYSTEM LABORATORY	CHP0604	4-14
DS - PROGRAM MANAGEMENT		75/05/21
4.0 PROGRAM COMMUNICATIONS 4.3 QUEUES	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	
	~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	
4.3.1 QUEUE REQUESTS		
The queue requests provided by follows:	Program Manager	ment are as
PM#ENQUEUE (queue, Item, status) PM#DEQUEUE (queue, Item, status) PM#STATUS_QJEUE (queue, status)	)	
4.3.1.1 <u>PM#ENQUEUE</u>		
The PM#ENQUEUE request adds activates one process if there is on	a queue ltem to ne suspended on th	ne queue.
PM#ENQUEUE (queue, item, status)	)	
queue: pointer to the queue con particular queue.	trol block which	defines the
item: pointer to the queue ite queue.	≥m which is to be	
status: returned request status	; •	
4.3.1.2 <u>PM#DEQUEUE</u>		
The PM#DEQUEUE request removes returns the location of the ltem to is empty at the time of the request,	the requestor.	If the queue suspended.
PM#DEQUEUE (queue, item, status)		:
queue: pointer to the queue co particular queue.	introl block which	
item: returned pointer to item longer on the queue.	data. The queue	
status: returned request status		
NCR/	CDC PRIVATE REV	30 APR 75

	DGRAM COMMUNICATIONS PM#STATUS_QUEUE		4.0 PROGRAM COMMUNICATIONS 4.3.1.3 PM#STATUS_QUEUE	~~~~~~~~~~
4.3.1.	3 PM#STATUS_QUEUE			
	ne PM#STATUS_QUEUE request provid are any items on the specified queu		4.4 <u>SEMAPHORES</u>	
РМ	∦STATUS_QUEUE (queue, status)		Semaphores are system supported facilitie:	
qu	eue: pointer to the queue control b queue.	lock that defines the 1 1	communication and synchronization among asynchro within a job. A semaphore is represented by a set block somewhere in storage and two system reques the control block. A companyous is the most point	maphore contro ts to manipulate
st	atus: Indicates whether or not th queue.		the control block. A semaphore is the most pri supported by the operating system for sync serialization of asynchronous activities. utilized by various system routines to serializ in the implementation of Locks and Queues.	hronization and Semaphores are
		1 1 1 1 2	A semaphore may be either an LNS variable or the Job data base (internal static, stack, etc. as shown below <b>:</b>	
		2 2 2 2 2	TYPE SEMAPHORE = RECORD VALUE: INTEGER, CHAIN: ~CONTROL_POINT,	
		22	RECEND;	
		2	ANY_SEMAPHORE: SEMAPHORE	
		3 3 3	The states of a semaphore are shown in the fo	)llowing table:
		3		
		3 3 4		
		4 4 4		
		4 4 4		

4-15

.....

4-16

•

GDS - PROGRAM MANAGEMENT		CHP060	4	4-17 75/05/21	ADVANCED SYSTEM LABORATORY CHP0604 IPLOS GUS - PROGRAM MANAGEMENT 75/05/21
4.0 PROGRAM COMMUNICATIONS 4.4 SEMAPHORES	~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~				4.0 PROGRAM COMMUNICATIONS 4.4.1.2 PM#WAIT_SEMAPHORE
+ REQUESTED OPERATION	+			+ NAL I	4.4.1.2 <u>PM#WAIT_SEMAPHORE</u>
: "VALUE" (V) +	<0   =0 ++	1 > 0 1 ++-	<0   = +	 0   >0   ++	This request decrements the 'value' of a semaphore by one If the resultant value is less than zero, the requesting proces is suspended.
l Resultant contents of	V-1 V-1	I V-1 I	V+1   V		PM#WAIT_SEMAPHORE (semaphore, status)
l I Add request process			i		semaphore; pointer to a structure of type semaphore
	• •				status: returned request status.
from thread and activate	• •				4.4.2 INTRA-JOB LOCKS
Process immediately   continues	     ++		x	X I. X I	Locks as such are not directly supported by the operating system as primitive requests since their function can be wholl
For a description of examples of their uses, Notes. 4.4.1 SEMAPHORE REQUESTS	see sectio				in Denning's article in the section titled Program Managemer Notes. A more flexible lock mechanism is proposed by the CODAS Programming Language Committee Proposal ATG-71001.11. See th section titled Program Management Notes.
The semaphore request follows:	s provided i	by Progra	m Manag	ement are as	4.4.3 INTER-JOB SYNCHRONIZATION
PM#SIGNAL_SEMAPHORE (s PM#WAIT_SEMAPHORE (sem					The semaphore and lock mechanisms described above are fo synchronization of asynchronous activities within a Job. Ther are two mechanisms which permit synchronization and communicatio <u>between</u> Jobs. One is the Signal facility described in 4.2. Th
	E				other is the Compare and Swap hardware instruction which may bused on memory locations which are shared between Jobs. Th Compare and Swap is externalized by two requests referencing
4.4.1.1 <u>PM#SIGNAL_SEMAPHOR</u>					signature lock. The two coordinating jobs must be sharing
4.4.1.1 <u>PM#SIGNAL SEMAPHOR</u> This request increm If the resultant value is which nas been waiting activated.	less than or	r equal t	o zero,	the process	segment with an agreed upon word in that segment designated a the signature lock.
This request increm If the resultant value is which nas been walting	less than or for the	r equal t semaphor	o zero,	the process	
This request increm If the resultant value is whicn nas been waiting activated.	less than or for the semaphore, s a structure	r equal t semaphor tatus) of type	o zero, e the	the process longest is	

1

2 3

4 5

6

7

8

15 16

17

18 19

20 21

22 23

28 29

30

31 32 33

34 35

36 37 38

75/05/21

TPLOS GUS - PROGRAM MANAGEMENT 

## 4.0 PROGRAM COMMUNICATIONS

4.4.3.1 Signature Lock Requests ST STANDICE LOCK REQUESTS

### PM#SIGN LOCK (lock, status) PM#UNSIGN LOCK (lock, status)

### 4.4.3.1.1 PM#SIGN LOCK

This request is used to sign a signature lock with the Control Point id of the requestor. The request is rejected if the requesting control point already has anything locked via PM#SIGN LOCK. Otherwise the request disables signal processing. 9 does a compare swap on the signature lock word. If the compare 10 swap is successful, returns leaving the Control Point id in the 11 signature lock word. If not successful, enables signal 12 processing and cycles. 13 14

CHRUEUT

PM#SIGN LOCK (lock, status)

lock: pointer to the signature lock word in the shared segment.

status: returned request status.

### 4.4.3.1.2 PM#UNSIGN_LOCK

This request is used to unsign a signature lock by writing 24 It with zeroes. Rejects if the requesting control point does not 25 have it locked. 26 27

# PM#UNSIGN_LOCK (lock, status)

lock; pointer to the signature lock word in the shared segment.

status: returned request status.

# 4.5 ON CONDITIONS

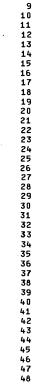
To be supplied.

NCR/CDC PRIVATE REV 02 AUG 74

ADVANCED SYS	TEM LABORATORY	CHP0 60 4	5-
	PROGRAM MANAGEMENT		75/05/2
 5.0	PROGRAM MAINTENANCE		
~~~~~			

.

lo Be Supplied



NCR/CDC PRIVATE REV 26 MAR 75

ADVANCED SYSTEM LABORATORY	CHP0 60 4	6-1
IPLOS GUS - PROGRAM MANAGEMENT	 	75/05/21

6.0 PROGRAM MANAGEMENT NOTES

6.0 PROGRAM MANAGEMENT NOTES

#### 6.1 COBOL LOCK NOTES

A flexible lock mechanism is proposed by the CODASYL Programming Language Committee Proposal ATG-71001.11. The lock defined there has four possible states and three operations defined on it as detailed in the following diagram:

REQUESTED OPERATION		U 	NL00	ск					FOR USE			LOCK			:
INITIAL STATE	i u	i	L	I M	1	Ε	U	I L	I M	I E	U	I L	I M	1	
Resultant	1	1			1		1	1	1	1	+ 1	1	+ 1	+	
state	ΙU	1	U	I M	Ł	U	I L	ΙΜ.	I M	ΙE	Ε.	I L	1 M	1 8	2
	1	1		:	ł		۱. J	1	1.	1	1	1	1	1	1
Suspend	1	1		1	1		1	1	1	1	1	1	1	1	i
requesting	1	- 1		1	1		1	1	1	1	1	1	1	1	
process	1	1	1	· ·	1		1	1	1	1 X	1	1 X	1 X	1 )	(
	1	1	1	E .	I.		1	1	1	1	1	1	1	1	1
Activate	1	1		1	1		1	1	1	1	1	1	1	1.	
suspended	1	1		1	1		1	1	1	1	1	1	1	1	
process	1	1	1		1		1	1	1	1	1	1	1	1	
if any	1	1	X	1.1	I.	X	1	1	:	:	1 1	1	1	1	
	1	1		L .	1		1	1	1	1	1	1	1	1	1
Request error	1 X	1		1	ŧ		1	1	1	1	1	1	1	1	
	1	1		1	1		1	1	1	1	1	1	1	1	
Process	1	1	1	i -	I.		ł	1	:	1	1	1	1	1	
immediately	1	1		1	1		1	1	:	1	1	1	1	1	1
continues	I X	1	X	<b>X</b> .	1	х	I X	I X	1 X	1	1 X	1	1	1	

# The states correspond to:

L = locked for shared use by one process.

M = locked for shared use by multiple processes.

U = unlocked.

NCR/CDC PRIVATE REV 26 MAR 75

# ADVANCED SYSTEM LABORATORY

IPLOS GDS - PROGRAM MANAGEMENT

6.0	PROGRAM	MA	NAGEMENT	NOTES
6.1	COBOL L	OCK	NOTES	

E = locked for exclusive use.

The following sample coding demonstrates how this fairly complex lock mechanism could be implemented by a run time support system or macros using the semaphore mechanism. However, this code example does not include the COBOL ATG 'AT LOCKED' immediate return option.

CHP0604

"Definition of a lock structure"

TYPE

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

42

43

44

45

46

47

48

COBOL_LOCK = RECORD . . SHARED COUNT : INTEGER EXCLUSIVE_LOCK : SENAPHORE, SHARED_KEY : SEMAPHORE, EXCLUSIVE_KEY : SEMAPHORE, RECEND;

"Semaphore used to serialize lock/unlock procedures"

LOCK_CONTROL : SEMAPHORE := [1, nIL];

"Unlock procedure used for both shared and" "exclusive locks"

PROC COBOL_UNLOCK (REF I: COBOL_LOCK); WAIT (LOCK_CONTROL); IF I.EXCLUSIVE_LOCK.VALUE LT 1 THEN "If exclusively locked" SIGNAL (I.SHARED_KEY); SIGNAL (I.EXCLUSIVE_LOCK); ELSE "LOCKED FOR SHARED USE" I. SHARED COUNT := I. SHARED_COUNT-1; IF I.SHARED_COUNT EQ O THEN "activate waiters for exclusive lock"

SIGNAL (I.EXCLUSIVE_KEY); IFEND; IFEND;

SIGNAL (LOCK_CONTROL); PROCEND COBOL_UNLOCK;

"Lock procedure for shared lock"

PROC COBOL_SHARED_LOCK (REF I: COBOL_LOCK); LABEL START_LOOP;

NCR/CDC PRIVATE REV 26 MAR 75

1

2

3

4

5

6 7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

DVANCED SYSTEM LABORATORY	CHP 0 60 4	6 <del>-</del> 3	ADVAN	ED SYSTEM LABORATORY	CHP 0 604	6-4
IPLOS GUS - PROGRAM MANAGEMENT		75/05/21		GDS - PROGRAM MANAGEMENT	75/05/21	
		******				
6.0 PROGRAM MANAGEMENT NOTES 6.1 Cobol Lock Notes				6.0 PROGRAM MANAGEMENT NOTES 6.1 Cobol Lock Notes		
SII COSCL LOCK NOTES		~~~~~~~~~~~		6.1 COBOL LUCK NOIES		
START_LOOP: LOOP;			1	IFEND;		
WAIT (LOCK_CONTROL);	·		2	IFEND;		
IF I.LXCLUSIVE_LOCK.VALUE EQ 1 T			3	LOOPEND;		
"If not exclusively locked"			4	PROCEND COBOL_EXCLUSIVE_LOCK.		
IF I.SHARED_COUNT = 0 THEN			5			
"If unlocked prevent exc	lusive lock"	· · ·	6			
WAIT (I.EXCLUSIVE_KEY);			7	6.2 <u>SEMAPHORE_NOIES</u>		
IFEND;			8			
I.SHARED_COUNT #= I.SHARED_C	COUNT + 1;		9			
SIGNAL (LOCK_CONTROL);			10	The following excerpt by De		
EXIT STARTLOOP;			11	the properties of semaphores ar	nd provides some exam	oles of their
ELSE			12	uses:		
"If exclusively locked wait	until unlocked"		13			
SIGNAL (LOCK_CONTROL);			14	"A semaphore is an i		
WAIT (I.SHARED_KEY);			15	value s0 ≥ 0 assigned on creatio		
SIGNAL (I.SHARED_KEY);			16	Q, in which are placed the ide		
IFEND;			17	the semaphore to be "unlocked."	Two indivisible op	erations are
LOOPEND;			18	defined on a semaphore s:		
PROCEND COBOL_SHARED_LOCK;			19			
•			20	wait s:[s<- s - 1; if s < 0		
			21	queue Q, enters the wait	ting state, and re	eleases th
			22	processor]		
			23			
			24			om Q, and add
"Lock procedure for exclusive lock"			25	it to the work queue of the	processors]	
	· ·		26			
PROC COBOL_EXCLUSIVE_LOCK (REF I: CO	BOL_LOCK);		27	Semaphore values may not be		
LABEL START_LOOP;			28	wait and signal operation. I		
START_LOOP &LOOP			29	processes waiting in the queue G		
WAIT (LOCK_CONTROL);			30	does not delay the caller, bu		
IF I.EXCLUSIVE_LOCK.VALUE LT 1 T			31	until another process executes a		
"If already exclusively lock	ed"		32	signal does not delay the ca	aller. The programmin	ng for mutua
SIGNAL (LOCK_CONTROL);			33	exclusion using wait and signal		
WAIT (I.EXCLUSIVE_LOCK);			34	unlock, with x0 = 1 (wait r	eplaces lock, and sig	gnal replaces
SIGNAL (I.EXCLUSIVE_LOCK);			35	unlock)		
ELSE		•	36			
IF I.SHARED_COUNT >0 THEN			37			
"If locked for shared us	e"		38	"Synchronization	·	
SIGNAL (LOCK_CONTROL);			39			
WAIT (I.EXCLUSIVE_KEY);			40	In a computation performed		
SIGNAL (I.EXCLUSIVE_KEY)	;	*	41	processes may not continue the		
ELSE			42	been supplied by others.		
"If unlocked"			43	program-executions proceed a		
WAIT (I.EXCLUSIVE_LOCK);			44	requirement that certain program	n-executions be ordere	ed in time.
"Prevent shared lock"			45			
PM#WAIT (I.SHARED_KEY);			46			
SIGNAL (LOCK_CONTROL);			47	1 Third Generation Computer Syst	ems", Peter J. Denni	ing, Computer
EXIT STARTLOOP:			48		per, 1971, pp. 199-20	

AD VANCED SYSTE	M LABORATORY	CHP0604

IPLOS GUS - PROGRAM MANAGEMENT 

6.0 PROGRAM MANAGEMENT NOTES 6.2 SEMAPHORE NOTES

This is called synchronization. The precedence constraints existing among processes in a system express the requirement for synchronization. Mutual exclusion is a form of synchronization in the sense that one process may be blocked until a signal is received from another. The wait and signal operations, which can pe used to express all forms of synchronizations, are often called synchronizing primitives.

"An interesting and important application of synchronization arises in conjunction with cooperating cyclic processes. An 10 example made famous by Dijkstra is the "producer/consumer" 11 problem, an abstraction of the input/output problem. Two cyclic 12 processes, the producer and the consumer, share a buffer of n > 013 cells, the producer places items there for later use by the 14 consumer. The producer might, for example, se a process that 15 generates output one line at a time, and the consumer a process 16 that operates the line printer. The producer must be blocked 17 from attempting to deposit an item into a full buffer, while the 18 consumer must be blocked from attempting to remove an item from 19 an empty buffer. Ignoring the details of producing, depositing, 20 removing, and consuming items, and concentrating solely on 21 synchronizing the two processes with respect to the conditions 22 "buffer full" and "buffer empty", we arrive at the following 23 abstract description of what is required. Let ala2...ak be a 24 system action sequence for the system consisting of the producer 25 and consumer processes. Let p(k) denote the number of times the 26 producer has deposited an item among the actions a1a2...ak, and 27 let c(k) denote the number of times the consumer has removed an 28 item from among the action a1a2...ak. It is required that 29

# $0 \leq p(k) - c(k) \leq n$ (1)

for all k. The programming that implements the required synchronization (Eq. 1) is given below; x and y are semaphores with initial values x0 = 0 and y0 = n:

pro:produce item;	con‡wait x;	
wait y;	remove item;	
deposit item;	signal y;	
signal x;	consume item;	
goto pro;	goto con;	

To prove that Eq. 1 holds for these processes, suppose 43 otherwise. The either c(k) > p(k) or p(k) > c(k) +n. However, 44 c(k) > p(k) is impossible since it implies that the number of 45 completed wait x exceeds the number of completed signal x, thus 46 contradicting x0 = 0. Similarly, p(k) > c(k) + n is also 47 impossible since it implies that the number of completed wait y 48

NCR/CDC PRIVATE REV 26 MAR 75

ADJANCED SYSTEM LABORATORY

6-5

1

2

3

4

5

6

q

30

31

32

33

34

35

36

37

75/05/21

75/05/21

CHP0 60 4

IPLOS GDS - PROGRAM MANAGEMENT

6.0 PROGRAM MANAGEMENT NOTES

6.2 SEMAPHORE NOTES

exceeds by more than n the number of completed signal y, thus contradicting y0 = n.

"Another application of synchronization is the familiar "read-acknowledge" form of signaling, as used in sending a message and waiting for a replay (B5). Define the semaphores r and a with initial values r0 = a0 = 0; the programming is of the form:

IN_IHE_SENDER	IN_IHE_RECEIVER
•	•
•	•
•	•
generate message;	wait r;
signal r;	obtain message;
wait a;	generate reply;
obtain reply;	signal a;
•	•
•	•

"...As a final example, let us consider how the 22 synchronizing primitives can be used to describe the operation of 23 an interrupt system. Typically, the interrupt hardware contains 24 a set of pairs of flipflops, each pair consisting of a "mask 25 flipflop" and an "interrupt flipflop." the states of the 26 flipflops in the ith pair are denoted by mi and xi, 27 respectively. The ith interrupt is said to be "disabled" (masked 28 off) if mi = 0, and "enabled" if mi = 1. When the hardware 29 senses the occurrence of the ith exceptional condition Ci, it 30 attempts to set xi = 1; if mI = 0, the setting of xi is delayed 31 until mi = 1. The setting of xi is supposed to awaken the ith 32 "interrupt-handler process" Hi, in order to act on the condition 33 Ci. By regarding mi and xi as hardware semaphores with initial 34 values mi = 1 and xi = 0, we can describe the foregoing 35 activities as an interprocess singaling problem: 36 37

	51
IN_HARDWARE	38
Ci occurstwait mi;	39
signal xi;	40
signal mi;	41
disable: wait mi;	42
enable: signal mi;	43
	44
IN_INTERRUPT_HANDLER_HI	45
start:wait xi:	46
process interrupt;	47
goto start;	48
-	

NCR/CDC PRIVATE REV 26 MAR 75

6-6

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21