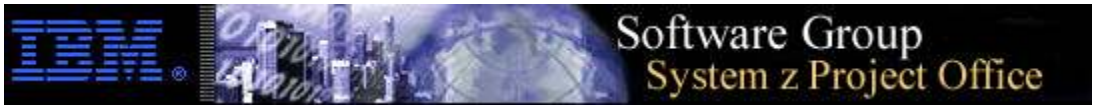
w3 COMP IBM.

Kettner's Corner #10 - zTidBits@CPO (Dispatchable UOW)



John Kettner is a member of the Z CPO team and teaches classes on z/OS fundamentals to customers and IBMers. John will share his expertise sharing his answers on key questions and topics on Z.

Check out the System z Project office Website: http://w3.ibm.com/sales/competition/compdlib.nsf/weball/9CF22F2F65E01639872571 OpenDocument

05/08/07 - #10 zTidBit@CPO (Dispatchable UOW)

Technical Rating: * (HIGH): 6+ years mainframe**
for this doc. ** (MEDIUM): 2- 6 years mainframe
(see BOLD) * (LOW): 1 - 2 year(s) mainframe

The last zINTRO Class asked about dispatchable units of work in z/OS. There is little basic information on this topic and when there is it's likely scattered in several publications.

Some background - The zOS Dispatcher is a routine within the nucleus (BCP) of the operating system that determines which units of work will gain control of the processor next. It maintains queues of dispatchable units of work each with an associated priority (dispatching priority) and whenever the dispatcher is given

control it selects the highest priority "ready" unit of work to dispatch.

There are two types of dispatchable units of work (UOW) that run within z/OS. The first one is called a Task Control Block (TCB) and the other is a Service Request Block (SRB).

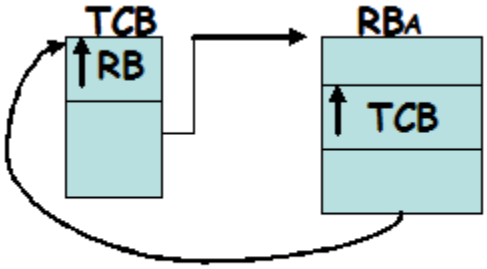
A TCB is a small structure in memory containing the current process status used by the address space representing a problem-state program (user program). It's a common form of managing the "bookkeeping" of units of work in real-time. As you should recall an address space in z/OS is a storage area (container) which houses your runtime environment. Requests to z/OS runs inside an address space (a.k.a. Region) and is represented by a structure in memory called an Address Space Control Block (ASCB).

A TCB dates back prior to MVS under a predecessor operating system known as MVT (Multi Programming with Variable Number of Tasks) used during the S360 and S370 era. This operating system eventually grew into MVS where the name "task" came from the MVT control block that was used to maintain information about a program running under MVT. Therefore, a task is any one unit of work running under an operating system. In most of these early operating systems a job only had one task. Today, multitasking is the term used to represent more than one task (and therefore TCBs) running inside a single job. A task is a unit of work to the system and competes for system resources. Therefore, a TCB represents a task and there is one TCB per task in the system. TCBs are used as a means of monitoring the phases of tasks and are located in the address space's private storage area. There are additional pointers (addresses) in a TCB pointing to other

control blocks used by program management, task management and the virtual storage manager used for TCB execution. It would be difficult to enumerate all of the significant pointers in this document, although suffice to say you can reference them in zOS' MVS Data Areas publication Volume 5 GA22-7585-08.

Task Control within z/OS can propagate. That is, one task may invoke another task and that task can invoke yet another task. Storage, scheduling and resource control are managed on a dynamic basis via a WLM Policy.

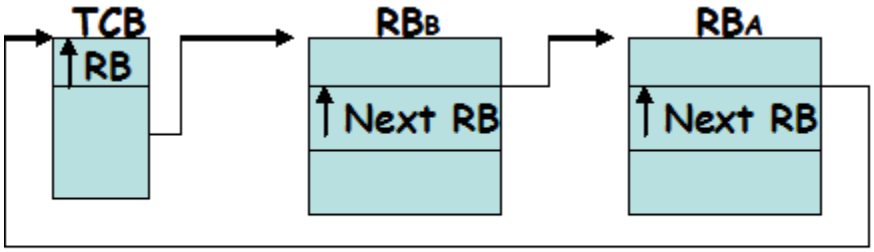
When executing a Program A,
i.e.
//MYJOB JOB
//STEP1 EXEC PGM=A *<-- creates TCB*
//DATAIN DD ….
//DATAOUT DD...
the initiator gives control to this program by a piece of special code called an ATTACH routine bringing the load module containing this program into virtual storage. The Program Manager who processes the ATTACH routine will create a TCB for this task. At this point the load module brought in is a level of control for the task and is represented by another control block known as a Request Block (RB). Since this is the only load module for this task there will be only one RB pointed to by the TCB. Program A is now active and the CPU is executing instructions in Program A.



Request Block chained to a Task Control Block

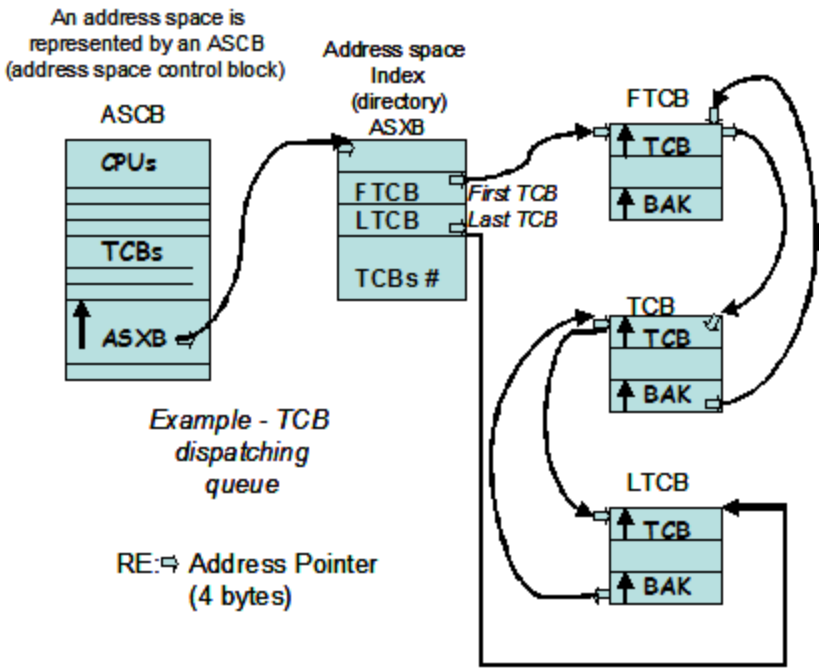
If Program A calls out to another Program

B (LINK routine) to execute supplemental logic it calls a supervisor routine (SVC) to request that Program B be loaded and begin execution. Now the RB structure has to change because of the new level of control by load module B. This activity is conceptualized here. The TCB will always have a pointer to the current RB representing the module being executed. The latest RB in the RB chain will always point back to the TCB.



Request Blocks chained after a CALL to PROGRAM B

Since an address space can contain many executing programs there is always more than one TCB associated with an address space. Therefore, there is a need to keep track of them and is done through two TCB queues. One is called the TCB dispatching queue or TCB READY Queue. The other is called Subtask Queue or family queue. The subtask queue can be considered providing the relationships via pointers (addresses) between TCBs or the hierarchy of TCBs.



An SRB represents a demand to execute a z/OS service routine on behalf of an end user's request. It is usually initiated by the system code executing in one address space to perform an action affecting the same or another address space. Thus a function in one address space can schedule an SRB to prompt a function in another address space to process some data place in common storage by the first function. As an example, when VTAM has received a message from a terminal via the I/O subsystem, it places the message in common storage, determines which address space it is intended for, then schedules an SRB into that address space to inform it that a message has been received and where it is located in storage.

SRBs are created using a different routine than a TCB ATTACH. They are initiated using a SCHEDULE routine (macro) and are run in a privileged state known as Supervisor State (executing in key 0).

There are two types of SRB: SRBs with global priority, which have a very high dispatching priority for inter-address space services and are given priorities above that of any address space, regardless of the

actual address space in which they are dispatched. The second type of SRBs has local priority and takes on the dispatch priority of the address space in which they are scheduled to run (intra-address space).

In general, SRBs are non-preemptive, which means if they are interrupted, control must be returned to them immediately after the interruption has been processed. SRBs usually represent very short pieces of work that complete much more quickly than TCBs.
NOTE: There are exceptions to this non-preemptive function and not covered in this document.

SRBs have the ability to SUSPEND or RESUME TCBs in their address space(s) and they can be scheduled by a function in one address space to execute in another address space such as providing a mechanism by which one address space can control the execution of tasks in another address space.

As seen in the TCB dispatching queues there are also chains of control blocks representing address spaces and units of work. The address space control block (ASCB) ready queue is a chain of ASCBs of those address spaces are swapped in and contain at least one TCB or SRB which is "ready" to execute. ASCBs are chained together in **priority order** (*SEE attached PPT from WLM Presentation below)* which the ASCB with the highest dispatching priority is at the front of the chain. Each address space then has its own chain of "ready" SRBs and/or TCBs pointed to from its ASCB. Whenever an event completes (i.e. I/O) which change the status of an address space the relevant z/OS function updates the dispatching queue to reflect it. Whenever control is returned to the dispatcher after a unit of work has terminated or been interrupted, it selects

the next highest priority unit of work that is ready to execute.

There are some *special* exit routines. for example, with hardware recovery which will always be dispatched before any other unit of work, although in most situations the dispatcher will first select any Global SRBs in priority order (Inter-address space) passing those requests to targeted address spaces, then it will select the ASCB at the front of the dispatcher queue - the one with the highest priority. Within that address space the Local "ready" SRBs are selected in priority order and if there are none of these, then any "ready" TCBs will be selected.

1 ---> READY Global SRBs (in priority order)
2 ---> READY Address Space " "
3 ---> READY Local SRBs " "
4 ---> READY TCBs " "

From recent zCPO WLM Conference. The example of Service Classes represent classification of address space priorities.

Ex_Disp_Priorities.pdf

If you have question for John, send him an email at: John H Kettner/New York/IBM.

For a direct link to the Z Project Office Website Click Here:http://w3.ibm.com/sales/competition/compdlib.nsf/weball/9CF22F2F65E0163987OpenDocument

---

Document details

Date: May 8, 2007

Content owner:

For IBMers only. Not intended for business partners or customers.

Report handling guidelines

Content provider:
IBM SWG Competitive Project Office

Feedback

Tag and save this page to your dogear favorites.

What is dogear ?