Wednesday, October 22, 2025, Amphitheater, 1:00 PM

Troubleshooting AST Code

2025 OpenVMS Bootcamp
Portsmouth Sheraton
Portsmouth, New Hampshire USA

Agenda –

- What is an Asynchronous System Trap (AST)?
- When are ASTs used?
- Most OpenVMS applications use a single Kernel thread
- Multiple Kernel thread applications require extra caution
- Common AST hazards

References

- OpenVMS System Service Manuals
- OpenVMS Programming Concepts Manual
- “Introduction to AST Programming”
  OpenVMS 2024 Bootcamp,
  http://www.rlgsc.com/openvms-bootcamp/2024/
  introduction-to-ast-progrmming.html

Agenda
References
Introduction
AST Basics
Generating ASTs
. . .

Introduction

What is an Asynchronous System Trap (AST)?

- A subroutine call triggered outside of the normal processing state
- Triggers are I/O, Timers, Locks, and other events
- Variables shared between process-level and AST-level are a hazard
- Documentation is not always clear and cogent
- ADAWI, Insert/Remove Queue Interlocked

ASTs are NOT *IX signal

- Queued
- No coalescing
- Not preemptible

AST Features

- Synchronization logically equivalent to IPL level synchronization in the VMS Executive WITHIN a single process.
- High Efficiency
- Fewer limits than Event Flags

AST Programming Concepts

- Non-interruptable by other ASTs at same or lesser Access Modes.
- FIFO Execution
- AST Entry is via an asynchronous(!), simulated, CALLS instruction

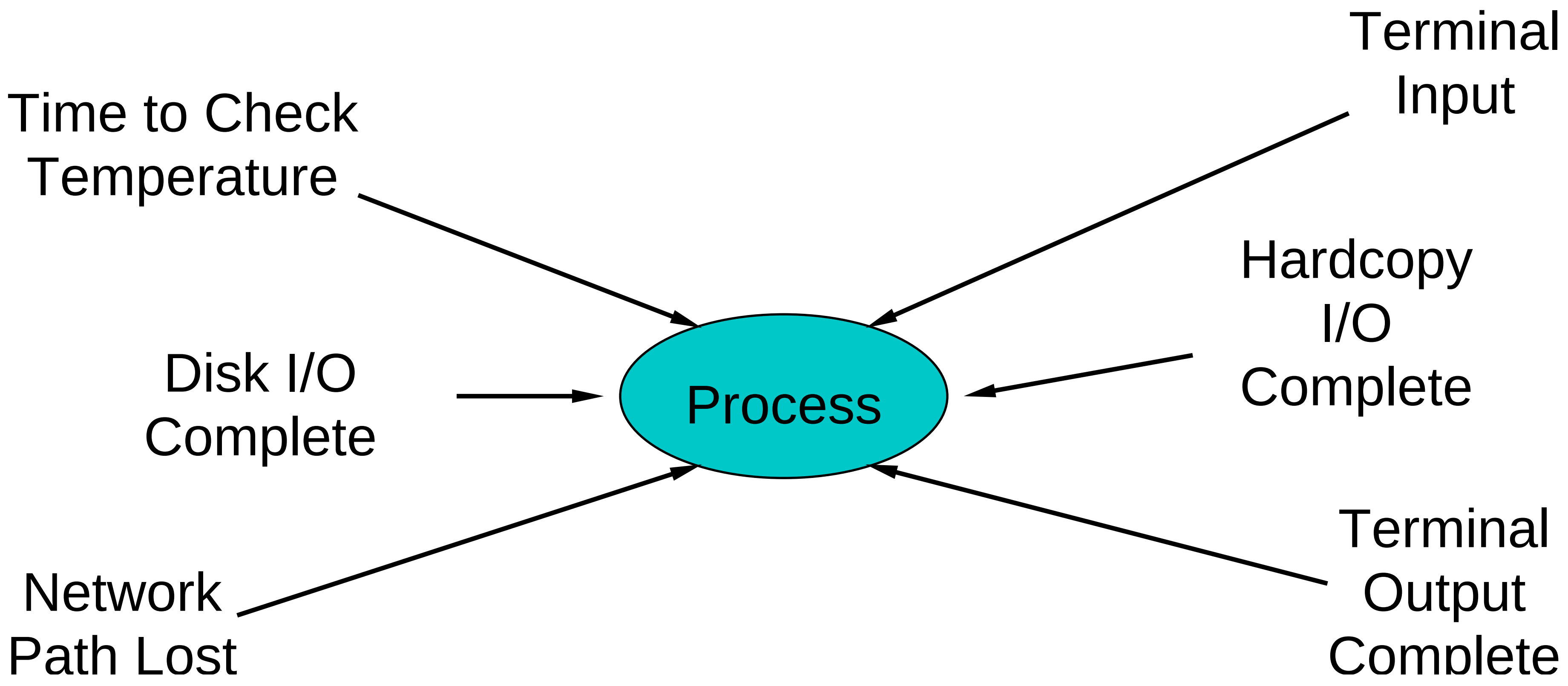
When are ASTs used?

- Realtime applications
- Control
- Transaction Processing
- Monitoring
- Network Applications
- Time–related applications

Typical Event Driven Computer Applications

- Printing
- Terminal Management
- Process Control

Typical Event Driven Computer Applications



Common Root – Events Determine Sequencing

- External Events control program
- Programs need to be efficient
- External event sequence is not under program control
- No dispatch routine

ASTs Creation

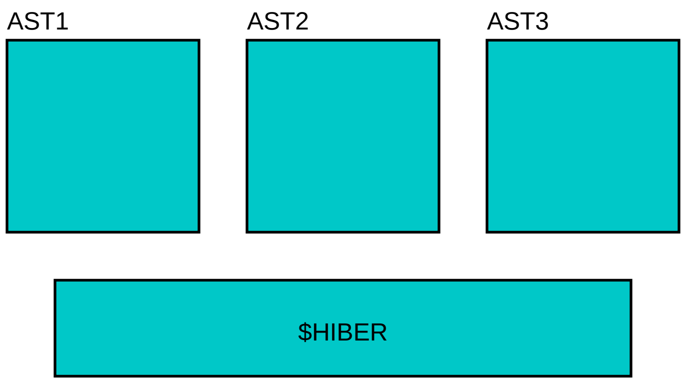
- $QIO
- $ENQ
- $TIMER
- Declare AST ($DCLAST)
- Mailboxes
- Unsolicited I/O Events
- Library Events

Speaking More Generally

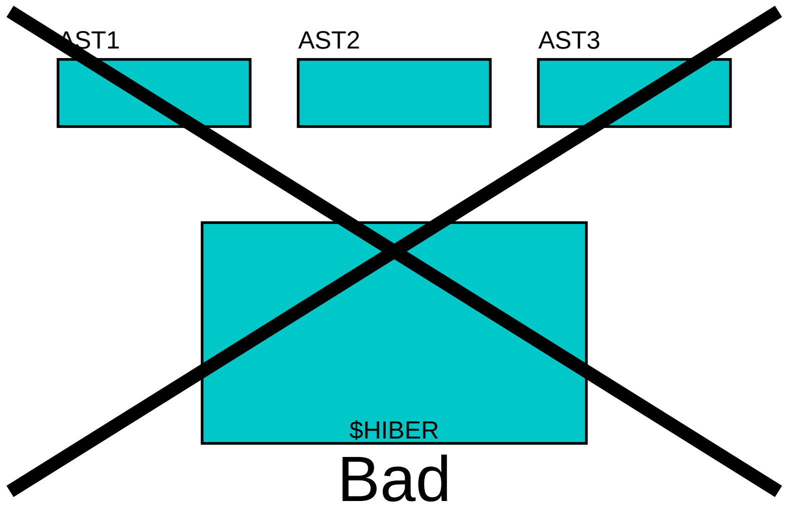
- Synchronous System Services (e.g., $FAO)
- Asynchronous System Services (e.g., $QIO
  – Descriptions include AST, ASTPRM, and IOSB)
- Derivatives thereof

Tricks to Getting It Right

- Do ALL Processing in ASTs.Do ALL Processing in ASTs.
- Avoid Performing Processing at AST level and normal Process level.



Good



Bad

Communications From AST Level To Process Level

- Use Queues, Insert/Remove Queue or LIB$ routines (for HLLs)
- Be careful of queue overflows, handle overflows gracefully
- Remember to ALWAYS issue $WAKE call!

Communications From Process Level to AST Level

- Use queues, Insert/Remove Queue or LIB$ routines (HLLs)
- Use $DCLAST service to switch to AST level
- Allow ASTs to be processed in the order they are queued. DO NOT process multiple items at a time!
- Processing sequence can be important

Avoid Problems

- Kill bugs before they occur
- DO NOT inhibit ASTs. Use $DCLAST to avoid interruptions
- Problems occur when realities are ignored
- Problems can be latent for long, unpredicatable periods
- When a problem is unresolved, set traps.

“Out of scope” Variables

- Asynchronous accessed variables must be STATIC, not DYNAMIC
- DYNAMIC variables are stored in the stack frame
- Stack frame disappears when routine exits
- Check IOSB, buffers whether local or parameters

Checking for AST State

- LIB$AST_IN_PROGRESS
- Caution: Invokes $GETJPI, LIB$GET_EF, LIB$FREE_EF
- Expensive, but possibly a check on errors

Enabling/Disabling AST Delivery

- $SETAST is dangerous; disabling AST delivery often has a code path that exits without re-enabling AST delivery
- Poor practice; classic bug is to bypass Enable
- If some processing needs interlocking – $DCLAST

Summary

- Anti-patterns often indicate problems waiting to happen
- Inhibiting AST is a hazardous practice
- If code is not AST-safe, consider entry gate assertion
- ASTs are perfectly safe when used correctly
-

. . .
Generating ASTs
Using ASTs
Generating and Processing
Hazards
Summary

Questions?

Robert Gezelter Software Consultant
35 – 20 167th Street, Suite 215
Flushing, New York 11358 – 1731
United States of America

+1 (718) 463 1079
gezelter@rlgsc.com
http://www.rlgsc.com

Session Notes & Materials:
http://www.rlgsc.com/openvms-bootcamp/2025/index.html