IBM

*7.1*

SC41-5606-09

IBM i

Conceitos de ILE

IBM

Conceitos do IBM i ILE

*7.1*

SC41-5606-09

**Nota**

Antes de usar essas informações e o produto que elas suportam, leia as informações no Apêndice D, "Avisos", na página 187.

Esta edição aplica-se ao IBM i 7.1 (número de produto 5770-SS1) e a todas as libertações e modificações subsequentes até indicação em contrário em novas edições. Esta versão não é executada em todos os modelos RISC (Reduced Instruction Set Computer) nem em modelos CISC.

**© Direitos autorais IBM Corporation 1997, 2010.**

Direitos Restritos de Usuários do Governo dos EUA – Uso, duplicação ou divulgação restrita pelo Contrato de Agendamento GSA ADP com a IBM Corp.

**Conteúdo**

**Sobre os Conceitos ILE (SC41-5606). . . . . . . . . . . . . . . . . . . . . . xi**

Quem deve ler este livro . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .xi

Pré-requisito e informações relacionadas . . . . . . . . . . . . . . . . . . . . . . . . . . .xi

Como enviar seus comentários . . . . . . . . . . . . . . . . . . . . . . . . . . . . .xi

**O que há de novo para o IBM i 7.1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . Xiii**

**Capítulo 1. Ambiente Linguístico Integrado Introdução . . . . . . . . . . . . . . . . . . . . . . . . . . 1**

O que é ILE? . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

Quais são os benefícios do ILE? . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

Ligação. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

Modularidade. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

Componentes reutilizáveis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .algarismo

Serviços do Common Runtime . . . . . . . . . . . . . . . . . . . . . . . . . . . . .algarismo

Coexistência com aplicativos existentes . . . . . . . . . . . . . . . . . . . . . . . . .algarismo

Depurador de origem . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .algarismo

Melhor controle sobre os recursos. . . . . . . . . . . . . . . . . . . . . . . . . . . . .algarismo

Caminho de Dados Abertos Compartilhado — Cenário . . . . . . . . . . . . . . . . . . . . . . . . . . 3

Controle de Compromisso — Cenário . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

Melhor controle sobre as interações linguísticas . . . . . . . . . . . . . . . . . . . . . . . . 4

Idiomas mistos — cenário . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

Melhor otimização de código. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

Qual é a história do ILE? . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

Descrição do modelo original do programa . . . . . . . . . . . . . . . . . . . . . . . . . . 6

Principais Características do OPM. . . . . . . . . . . . . . . . . . . . . . . . . . . 6

Descrição do modelo de programa estendido . . . . . . . . . . . . . . . . . . . . . . . . . . 7

Descrição do Ambiente Linguístico Integrado . . . . . . . . . . . . . . . . . . . . . . . 7

Principais Características das Linguagens Baseadas em Procedimentos . . . . . . . . . . . . . . . . . . . 7

**Capítulo 2. Conceitos Básicos da ILE . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9**

Estrutura de um Programa ILE . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

Procedimento . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

Objeto Module . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

Programa ILE . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

Programa de Serviço . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

Diretório de vinculação . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

Vinculação de Processamento de Diretório . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

Funções do fichário . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

Chamando um Programa ou um Procedimento . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

Chamadas dinâmicas de programas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

Chamadas de procedimento estático . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

| Chamadas de ponteiro de procedimento . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

Ativação. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

Visão geral sobre o tratamento de erros . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

Otimizando o Tradutor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

Depurador. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

**Capítulo 3. ILE Advanced Concepts. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .**

Ativação do Programa. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23

Criação de Ativação de Programas . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

Grupo de ativação . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

Criação de Grupo de Ativação . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26

Grupos de ativação padrão . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27

Exclusão de grupo de ativação não padrão . . . . . . . . . . . . . . . . . . . . . . . . . 28

© Direitos autorais IBM Corp. 1997, 2010

Ativação do Programa de Serviço . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30

Limites de controle. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32

| Limites de Controle para Grupos de Ativação . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32 Limites de controle entre as entradas OPM e ILE Call Stack . . . . . . . . . . . . . . . . . . . . . . . 33 anos

Controle o uso do limite. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

Tratamento de erros . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34 Filas de mensagens de emprego . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35

Mensagens de exceção e como elas são enviadas . . . . . . . . . . . . . . . . . . . . . . . 35

Como as mensagens de exceção são manipuladas . . . . . . . . . . . . . . . . . . . . . . . . . 36

Recuperação de exceção. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

Ações padrão para exceções não tratadas . . . . . . . . . . . . . . . . . . . . . . . . 37

Tipos de manipuladores de exceção . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38

Condições ILE . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40

Regras de Escopo de Gerenciamento de Dados . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41 Escopo de nível de chamada . . . . . . . . . . . . . . . . . . . . . 41

Escopo de nível de grupo de ativação . . . . . . . . . . . . . . . . . . . . . . . . . . . 42

Escopo de nível de trabalho . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43

**Capítulo 4. Teraspace e armazenamento de nível único . . . . . . . . . . . . . . . . . . . . . . . . . . . 45**

Características do Teraspace . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45

Usando o Teraspace para armazenamento . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45

Escolhendo um Modelo de Armazenamento de Programas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46 Especificação do modelo de armazenamento Teraspace . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46

Selecionando um Grupo de Ativação Compatível . . . . . . . . . . . . . . . . . . . . . . . . 47

Como os modelos de armazenamento interagem . . . . . . . . . . . . . . . . . . . . . . . . . . . 47

Regras para módulos de vinculação . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47

Regras para vinculação a programas de serviço . . . . . . . . . . . . . . . . . . . . . . . . 48

Regras para Ativação de Programas e Programas de Serviço . . . . . . . . . . . . . . . . . . . 48

Regras para Chamadas de Programas e Procedimentos. . . . . . . . . . . . . . . . . . . . . . . . 48

Convertendo seu programa ou programa de serviço para herdar um modelo de armazenamento. . . . . . . . . . . . . . 48

Atualizando seus programas: Considerações sobre o Teraspace . . . . . . . . . . . . . . . . . . . . . 49

Aproveitando os ponteiros de 8 bytes em seu código C e C++ . . . . . . . . . . . . . . . . . . . . . . . . . . 49 Suporte de ponteiro em compiladores C e C++ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49

Conversões de ponteiro . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50

Usando o Modelo de Armazenamento Teraspace . . . . . . . . . . . . . . . . . . . . . . . . . . . 50

Usando o Teraspace: Práticas recomendadas . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51

Controles do sistema sobre programas Teraspace quando eles são criados. . . . . . . . . . . . . . . 51

Controles do sistema sobre programas Teraspace quando eles são ativados . . . . . . . . . . . . . . 51

IBM i Interfaces e Teraspace . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51

Problemas potenciais que podem surgir quando você usa o Teraspace . . . . . . . . . . . . . . . . . . 52

Dicas de uso do Teraspace . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52

**Capítulo 5. Conceitos de Criação do Programa. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55**

Criar Comandos de Programa e Criar Programa de Serviço . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55 Utilizar a Autoridade Adoptada (QUSEADPAUT) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56

Usando parâmetros de otimização . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56

Dados armazenados em módulos e programas . . . . . . . . . . . . . . . . . . . . . . . . . . 56

Resolução do símbolo . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57 Importações resolvidas e não resolvidas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57

Vinculação por cópia . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58

Vinculação por Referência . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58

Vinculação de um grande número de módulos . . . . . . . . . . . . . . . . . . . . . . . . . . 58

| Símbolos duplicados . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59

Importância da Ordem das Exportações . . . . . . . . . . . . . . . . . . . . . . . . . . 59

Exemplo 1 de criação de programa . . . . . . . . . . . . . . . . . . . . . . . . . . . 60

Exemplo de Criação de Programa 2 . . . . . . . . . . . . . . . . . . . . . . . . . . . 62

Acesso ao Programa . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64

Parâmetro do módulo de procedimento de entrada do programa no comando CRTPGM . . . . . . . . . . . . . 64

Parâmetro Export no comando CRTSRVPGM . . . . . . . . . . . . . . . . . . . . . 65

Parâmetro de exportação usado com os parâmetros do arquivo de origem e do membro de origem. . . . . . . . . . . . . 66

Conceitos de ILE IBM i 7.1

Largura máxima de um arquivo para o parâmetro SRCFILE . . . . . . . . . . . . . . . . . . . 66

Conceitos de Importação e Exportação. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66

Linguagem do fichário . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67

Assinatura. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68

Inicie os comandos Exportar Programa e Encerrar Exportação de Programa . . . . . . . . . . . . . . . . . . 69

Parâmetro de nível de programa no comando STRPGMEXP . . . . . . . . . . . . . . . . . . 69

Parâmetro Signature no comando STRPGMEXP . . . . . . . . . . . . . . . . . . . 69

Parâmetro Level Check no comando STRPGMEXP . . . . . . . . . . . . . . . . . . 70

Comando Exportar símbolo . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70

Exemplos de símbolos de exportação curinga . . . . . . . . . . . . . . . . . . . . . . . . . 70

Exemplos de linguagem de fichário . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71

Exemplo 1 da linguagem do fichário . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71

Exemplo 2 da linguagem do fichário . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72

Exemplo 3 da linguagem do fichário . . . . . . . . . . . . . . . . . . . . . . . . . . . . 74

Exemplo 4 da linguagem do fichário . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75

Alterando programas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78

Atualizações do Programa . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79

Parâmetros nos comandos UPDPGM e UPDSRVPGM . . . . . . . . . . . . . . . . . . 80

Módulo substituído por um módulo com menos importações. . . . . . . . . . . . . . . . . . . . . 81

Módulo substituído por um módulo com mais importações. . . . . . . . . . . . . . . . . . . . . 81

Módulo substituído por um módulo com menos exportações. . . . . . . . . . . . . . . . . . . . . 82

Módulo substituído por um módulo com mais exportações . . . . . . . . . . . . . . . . . . . . . 82

Dicas para criar módulos, programas e programas de serviço . . . . . . . . . . . . . . . . . . . 82

**Capítulo 6. Gestão de Grupos de Ativação. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85**

Vários aplicativos em execução no mesmo trabalho . . . . . . . . . . . . . . . . . . . . . . . 85

Comando Recuperar Recursos . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86

Comando Recuperar Recursos para Programas OPM . . . . . . . . . . . . . . . . . . . . . 88

Comando Recuperar Recursos para Programas ILE . . . . . . . . . . . . . . . . . . . . . . 88

Comando Recuperar Grupo de Ativação. . . . . . . . . . . . . . . . . . . . . . . . . . 88

Programas de Serviço e Grupos de Ativação . . . . . . . . . . . . . . . . . . . . . . . . . 88

**Capítulo 7. Convites à Apresentação de Procedimentos e Programas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91**

Pilha de chamadas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91

Exemplo de pilha de chamadas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91

Chamadas para Programas e Chamadas para Procedimentos. . . . . . . . . . . . . . . . . . . . . . . . . 92

Chamadas de procedimento estático . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93

Chamadas de ponteiro de procedimento . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93

Passando argumentos para procedimentos ILE . . . . . . . . . . . . . . . . . . . . . . . . . 93

Resultados da função. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94

Argumentos omitidos . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 95

Chamadas dinâmicas de programas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 95

Passando argumentos em uma chamada de programa dinâmico . . . . . . . . . . . . . . . . . . . . . . 96

Compatibilidade de dados interlinguísticos . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96

Sintaxe para passar argumentos em aplicativos de linguagem mista . . . . . . . . . . . . . . . . . 96

Descritores Operacionais . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96

Requisitos dos Descritores Operacionais . . . . . . . . . . . . . . . . . . . . . . . 97

ausência de um descritor obrigatório . . . . . . . . . . . . . . . . . . . . . . . . . . 97

Presença de um descritor desnecessário . . . . . . . . . . . . . . . . . . . . . . . . 97

APIs vinculáveis para acesso ao descritor operacional. . . . . . . . . . . . . . . . . . . . . 97

Suporte para APIs OPM e ILE. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 97

**Capítulo 8. Gestão de armazenamento. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 99**

Pilha de armazenamento de nível único . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 99

Características da pilha . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 99

Pilha padrão . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 99

Heaps criados pelo usuário . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 100

Suporte a pilha única. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 100

Estratégia de alocação de heap . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 101

Interfaces de heap de armazenamento de nível único . . . . . . . . . . . . . . . . . . . . . . . . . 101

Conteúdo

[Suporte a pilha 102](#_Toc502808)

[Armazenamento local de thread 102](#_Toc502809)

[**Capítulo 9. Gerenciamento de Exceções e** **Condições 105**](#_Toc502810)

[Manipular cursores e retomar cursores 105](#_Toc502811)

[Ações do manipulador de exceções 106](#_Toc502812)

[Como retomar o processamento 106](#_Toc502813)

[Como Percolar uma Mensagem 107](#_Toc502814)

[Como promover uma mensagem 107](#_Toc502815)

[Ações padrão para exceções não tratadas 108](#_Toc502816)

[Exceções aninhadas 109](#_Toc502817)

[Tratamento de Condição 109](#_Toc502818)

[Como as condições são representadas 109](#_Toc502819)

[Layout de um Token de Condição 110](#_Toc502820)

[Teste de token de condição 111](#_Toc502821)

[Relação das condições ILE com as mensagens do sistema operacional 111](#_Toc502822)

[Mensagens do IBM i e o Código de Feedback da API Bindable 111](#_Toc502823)

[**Capítulo 10. Considerações sobre depuração** **113**](#_Toc502824)

[Modo de depuração 113](#_Toc502825)

[Ambiente de depuração 113](#_Toc502826)

[Adição de programas ao modo de depuração 113](#_Toc502827)

[Como a observabilidade e a otimização afetam a depuração 114](#_Toc502828)

[Níveis de otimização 114](#_Toc502829)

[Depurar Criação e Remoção de Dados 114](#_Toc502830)

[Visualizações do módulo 114](#_Toc502831)

[Depuração entre Trabalhos 115](#_Toc502832)

[Suporte a depuradores OPM e ILE 115](#_Toc502833)

[Suporte do Watch 115](#_Toc502834)

[Exceções não monitoradas 115](#_Toc502835)

[Restrição de globalização para depuração 115](#_Toc502836)

[**Capítulo 11. Escopo de gerenciamento de dados** **117**](#_Toc502837)

[Recursos comuns de gerenciamento de dados 117](#_Toc502838)

[Escopo de Controle de Compromisso 118](#_Toc502839)

[Definições de Compromisso e Grupos de Ativação 118](#_Toc502840)

[Acabando com o Controle de Compromisso 119](#_Toc502841)

[Controle de Compromisso durante o Grupo de Ativação Final 120](#_Toc502842)

[**Capítulo 12. Interfaces de programação de aplicativos vinculáveis ILE** **121**](#_Toc502843)

[APIs vinculáveis ILE disponíveis 121](#_Toc502844)

[APIs vinculáveis do Gerenciador de Tela Dinâmica 123](#_Toc502845)

[**Capítulo 13. Técnicas avançadas de otimização** **125**](#_Toc502846)

[Perfil do programa 125](#_Toc502847)

[Tipos de criação de perfil 125](#_Toc502848)

[Como criar o perfil de um programa 126](#_Toc502849)

[Habilitando o programa para coletar dados de criação de perfil 126](#_Toc502850)

[Coletar dados de criação de perfil 127](#_Toc502851)

[Aplicando os dados de criação de perfil coletados 127](#_Toc502852)

[Gerenciando programas habilitados para coletar dados de criação de perfil 128](#_Toc502853)

[Gerenciando programas com dados de criação de perfil aplicados a eles 129](#_Toc502854)

[Como saber se um programa ou módulo está com perfil ou habilitado para a coleção 130](#_Toc502855)

[Análise interprocessual (IPA) 130](#_Toc502856)

[Como otimizar seus programas com IPA 132](#_Toc502857)

[Sintaxe do arquivo de controle IPA 132](#_Toc502858)

[Notas de uso do IPA 134](#_Toc502859)

[Restrições e limitações do IPA 134](#_Toc502860)

[Partições criadas por IPA 135](#_Toc502861)

[Otimização avançada de argumentos 136](#_Toc502862)

Conceitos de ILE IBM i 7.1

Como usar a otimização avançada de argumentos . 136

Considerações e restrições ao usar a otimização avançada de argumentos . 136

Opções de código interno licenciado . . . . . . . . . . . . . . . . . . . . . . . . . . . . 138

Opções definidas no momento . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 138

Aplicação. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 141

Restrições. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 141

Sintaxe. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 141

Compatibilidade de versão . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 142

Exibindo opções de código interno licenciado do módulo e do programa ILE . . . . . . . . . . . . . . 142

Geração de código adaptável . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 143

Conceitos ACG . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 143

Operação Normal. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 144

Opções de restauração . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 144

QFRCCVNRST Valor do Sistema . . . . . . . . . . . . . . . . . . . . . . . . . . . . 144

FRCOBJCVN Parâmetro . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 145

Criar opções . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 145

O CodeGenTarget LICOPT . . . . . . . . . . . . . . . . . . . . . . . . . . . . 146

QIBM\_BN\_CREATE\_WITH\_COMMON\_CODEGEN Variável de Ambiente . . . . . . . . . . . . . 147

Exibindo informações do ACG . . . . . . . . . . . . . . . . . . . . . . . . . . . . 147

Compatibilidade de objetos . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 147

Considerações de liberação para versão . . . . . . . . . . . . . . . . . . . . . . . . . . 148

Otimizando programas compatíveis . . . . . . . . . . . . . . . . . . . . . . . . . . 148

ACG e partições lógicas . . . . . . . . . . . . . . . . . . . . . . . . . . . . 149

**Capítulo 14. Sincronização de Armazenamento Compartilhado. . . . . . . . . . . . . . . . . . . . . . 151**

Armazenamento compartilhado . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 151

Armadilhas de armazenamento compartilhado . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 151

Pedido de acesso ao armazenamento compartilhado . . . . . . . . . . . . . . . . . . . . . . . . . . . . 152

Exemplo Problema 1: Um Escritor, Muitos Leitores . . . . . . . . . . . . . . . . . . . . . . 152

Exemplo 1 Solução . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 153

Ações de sincronização de armazenamento . . . . . . . . . . . . . . . . . . . . . . . . . . . . 153

Exemplo de Problema 2: Dois Escritores ou Leitores em Conflito. . . . . . . . . . . . . . . . . . . 154

Exemplo 2 Solução . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 154

Solução alternativa: Usando o valor de bloqueio de verificação / o valor de bloqueio de limpeza . . . . . . . . . . . . . . . 155

**Apêndice A. Listagem de saída do comando CRTPGM, CRTSRVPGM, UPDPGM ou UPDSRVPGM . 157**

Listagem do fichário . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 157

Listagem básica. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 157

Listagem estendida . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 159

Lista Completa . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 161

Componentes de listagem do IPA. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 163

Mapa do Arquivo de Objeto . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 163

Mapa de Opções do Compilador . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 163

Relatório embutido . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 163

Mapa de símbolos globais . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 164

Mapa de partição . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 164

Mapa do arquivo de origem . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 164

Mensagens. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 164

Resumo da mensagem . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 164

Listagem, por exemplo, do programa de serviço . . . . . . . . . . . . . . . . . . . . . . . . . 165

Listagem de informações do fichário, por exemplo, programa de serviço . . . . . . . . . . . . . . . . . . 165

Listagem de referência cruzada para o programa de serviço de exemplo . . . . . . . . . . . . . . . . . . . 165

Estatísticas de Vinculação, por exemplo, Programa de Serviço . . . . . . . . . . . . . . . . . . . . . 166

Erros de idioma do fichário . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 166

Assinatura acolchoada . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 167

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 167

Assinatura truncada . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 167

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 167

Interface atual de limites de bloco de exportação . . . . . . . . . . . . . . . . . . . . . . . . . 167

Conteúdo

Alterações sugeridas . . . . . . . . . 168 anos

Bloco de exportação duplicado . . . . . . . . . . . . . . . . . . . . 168 anos

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 169

Símbolo duplicado na exportação anterior . . . . . . . . . . . . . . . . . . . . . . . . . 169

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 170

A verificação de nível não pode ser desabilitada mais de uma vez, ignorada . . . . . . . . . . . . . . . . . 170

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 170

Vários blocos de exportação atuais não permitidos, assumidos anteriormente . . . . . . . . . . . . . . . . 170

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 171

Bloco de exportação atual está vazio . . . . . . . . . . . . . . . . . . . . . . . . . . . 171

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 171

Bloco de exportação não concluído, fim de arquivo encontrado antes do ENDPGMEXP . . . . . . . . . . . . . . 172

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 172

Bloco de exportação não iniciado, STRPGMEXP necessário . . . . . . . . . . . . . . . . . . . . . 172

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 173

Blocos de exportação não podem ser aninhados, ENDPGMEXP ausente . . . . . . . . . . . . . . . . . . . 173

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 173

As exportações devem existir dentro dos blocos de exportação . . . . . . . . . . . . . . . . . . . . . . . . 173

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 174

assinaturas idênticas para blocos de exportação diferentes, devem alterar as exportações. . . . . . . . . . . . . . 174

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 174

Várias correspondências curinga. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 174

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 175

Nenhum bloco de exportação atual . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 175

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 175

Sem correspondências curinga . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 175

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 176

O bloco de exportação anterior está vazio . . . . . . . . . . . . . . . . . . . . . . . . . . . 176

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 176

A assinatura contém caracteres variantes . . . . . . . . . . . . . . . . . . . . . . . . . 177

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 177

ASSINATURA(\*GEN) Necessária com LVLCHK(\*NO). . . . . . . . . . . . . . . . . . . . . 177

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 177

Sintaxe de assinatura não válida. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 178

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 178

Nome do símbolo necessário . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 178

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 178

Símbolo não permitido como exportação do programa de serviço . . . . . . . . . . . . . . . . . . . . . 179

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 179

Símbolo não definido . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 179

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 180

Sintaxe não válida . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 180

Alterações sugeridas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 180

**Apêndice B. Exceções em Programas Otimizados . . . . . . . . . . . . . . . . . . . . . . . . . . . 181**

**Apêndice C. Comandos CL Usados com Objetos ILE . . . . . . . . . . . . . . . . . . . . . . . . . . 183**

Comandos CL usados com módulos . . . . . . . . . . . . . . . . . . . . . . . . . . . 183

Comandos CL usados com objetos de programa . . . . . . . . . . . . . . . . . . . . . . . . 183

Comandos CL usados com programas de serviço . . . . . . . . . . . . . . . . . . . . . . . . 183

Comandos CL usados com diretórios de vinculação . . . . . . . . . . . . . . . . . . . . . . . 184

Comandos CL usados com a linguagem de consulta estruturada . . . . . . . . . . . . . . . . . . . . 184

Comandos CL usados com o CICS . . . . . . . . . . . . . . . . . . . . . . . . . . . . 184

Comandos CL usados com o depurador de origem . . . . . . . . . . . . . . . . . . . . . . . . 184

Comandos CL usados para editar o arquivo de origem do idioma do fichário . . . . . . . . . . . . . . . . . . 185

**Apêndice D. Comunicações . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 187**

Informações da interface de programação . . . . . . . . . . . . . . . . . . . . . . . . . . . 188

Marcas comerciais. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 189

Conceitos de ILE IBM i 7.1

**Bibliografia................................. 191**

**Índice.................................... 193**

Conteúdo

**x** Conceitos ILE IBM i 7.1

**Sobre os conceitos de ILE (SC41-5606)**

Este livro descreve conceitos e terminologia relativos à arquitetura do Integrated Language Environment (ILE) do programa licenciado do IBM® i. Os tópicos abordados incluem criação de módulos, vinculação, manipulação de mensagens, execução e depuração de programas e tratamento de exceções.

Os conceitos descritos neste livro pertencem a todas as linguagens ILE. Cada linguagem ILE pode implementar a arquitetura ILE de forma um pouco diferente. Para determinar exatamente como cada linguagem habilita os conceitos descritos aqui, consulte o guia do programador para essa linguagem ILE específica.

Este livro também descreve as funções do IBM i que pertencem diretamente a todas as linguagens ILE.

Este livro não descreve a migração de uma linguagem IBM i existente para uma linguagem ILE. Essas informações estão contidas no guia de cada programador de linguagem de alto nível (HLL) da ILE.

**Quem deve ler este livro**

Você deve ler este livro se:

v Você é um fornecedor de software que desenvolve aplicativos ou ferramentas de software.

v Tem experiência no desenvolvimento de aplicações de linguagem mista no sistema operativo IBM i. v Você não está familiarizado com o IBM i, mas tem experiência em programação de aplicativos em outros sistemas.

v Seus programas compartilham procedimentos comuns e, quando você atualiza ou aprimora esses procedimentos, precisa recriar os programas que os utilizam.

Além deste livro, certifique-se de verificar os guias do programador para as linguagens específicas em que você escreve programas.

**Pré-requisito e informações relacionadas**

Utilize o Centro de Informações do IBM i como ponto de partida para obter informações técnicas do IBM i.

Você pode acessar o centro de informações a partir do seguinte site:

http://www.ibm.com/systems/i/infocenter/

O Centro de Informações do IBM i contém informações do sistema novas e atualizadas, como instalação de software, Linux, WebSphere®, Java™, alta disponibilidade, banco de dados, partições lógicas, comandos CL e interfaces de programação de aplicativos (APIs) do sistema. Além disso, ele fornece consultores e localizadores para ajudar no planejamento, solução de problemas e configuração do hardware e software do sistema.

A cada novo pedido de hardware, você recebe o *DVD System i Access para Windows*, SK3T-4098. Este DVD fornece a instalação do programa licenciado IBM System i® Access for Windows. A família System i Access oferece recursos de cliente e servidor para conectar PCs a modelos IBM i.

Para outras informações relacionadas, consulte a "Bibliografia" na página 191.

**Como enviar seus comentários**

Seu feedback é importante para ajudar a fornecer as informações mais precisas e de alta qualidade. Se tiver quaisquer comentários sobre este livro ou qualquer outra documentação do IBM i, preencha o formulário de comentários dos leitores no verso deste livro.

v Se você preferir enviar comentários por correio, use o formulário de comentário dos leitores com o endereço impresso no verso. Se você estiver enviando um formulário de comentário de leitores de um país ou região diferente dos Estados Unidos, poderá fornecer o formulário à filial local da IBM ou ao representante da IBM para envio pago por postagem. v Se preferir enviar comentários por FAX, utilize um dos seguintes números:

* Estados Unidos, Canadá e Porto Rico: 1-800-937-3430
* Outros países ou regiões: 1-507-253-5192 v Se você preferir enviar comentários eletronicamente, use um destes endereços de e-mail:
* Comentários sobre livros:

RCHCLERK@us.ibm.com

* Comentários sobre o Centro de Informações do IBM i: RCHINFOC@us.ibm.com

Certifique-se de incluir o seguinte:

v O nome do livro ou do tópico do Centro de Informações do IBM i.

v O número de publicação de um livro. v O número da página ou tópico de um livro ao qual seu comentário se aplica.

**xii** Conceitos ILE IBM i 7.1

**O que há de novo no IBM i 7.1**

Aqui estão as principais mudanças nessas informações para esta edição.

| v Um novo grupo de ativação padrão teraspace foi adicionado.

| v ILE RPG e ILE COBOL agora suportam uma opção de modelo de armazenamento teraspace.

**xiv** Conceitos ILE IBM i 7.1

**Capítulo 1. Introdução ao Ambiente Linguístico Integrado**

Este capítulo define o modelo de Ambiente de Linguagem Integrado (ILE), descreve os benefícios do ILE e explica como o ILE evoluiu dos modelos de programa anteriores.

**O que é ILE?**

ILE é um conjunto de ferramentas e suporte de sistema associado projetado para melhorar o desenvolvimento de programas no sistema operacional IBM i.

Os recursos desse modelo podem ser usados somente por programas produzidos pela família de compiladores ILE. Essa família inclui ILE RPG, ILE COBOL, ILE C, ILE C++ e ILE CL.

**Quais são os benefícios do ILE?**

O ILE oferece inúmeros benefícios em relação aos modelos de programas anteriores. Esses benefícios incluem associação, modularidade, componentes reutilizáveis, serviços de tempo de execução, coexistência e um depurador de origem. Eles também incluem melhor controle sobre os recursos, melhor controle sobre as interações de linguagem, melhor otimização de código, um melhor ambiente para C e uma base para o futuro.

**Ligação**

O benefício da vinculação é que ela ajuda a reduzir a sobrecarga associada às operações de chamada. Unir os módulos acelera a chamada. O mecanismo de chamada anterior ainda está disponível, mas também há uma alternativa mais rápida. Para diferenciar entre os dois tipos de chamadas, o método anterior é chamado de chamada de programa dinâmico ou externo e o método ILE é chamado de chamada de procedimento estático ou acoplado.

A capacidade de ligação, juntamente com a melhoria resultante no desempenho das chamadas, torna muito mais prático desenvolver aplicações de uma forma altamente modular. Um compilador ILE não produz um programa que pode ser executado. Em vez disso, ele produz um objeto de módulo (\*MODULE) que pode ser combinado (ligado) com outros módulos para formar uma única unidade executável; ou seja, um objeto de programa (\*PGM).

Assim como você pode chamar um programa de RPG de um programa COBOL, o ILE permite que você vincule módulos escritos em diferentes idiomas. Portanto, é possível criar um único programa executável que consiste em módulos escritos separadamente em RPG, COBOL, C, C++ e CL.

**Modularidade**

Os benefícios do uso de uma abordagem modular para a programação de aplicativos incluem o seguinte: v Tempo de compilação mais rápido

Quanto menor o pedaço de código que compilamos, mais rápido o compilador pode processá-lo. Esse benefício é particularmente importante durante a manutenção, porque muitas vezes apenas uma linha ou duas precisam ser trocadas. Quando alteramos duas linhas, talvez tenhamos que recompilar 2000 linhas. Não se trata de uma utilização eficiente dos recursos.

Se modularizarmos o código e aproveitarmos os recursos de vinculação do ILE, talvez precisemos recompilar apenas 100 ou 200 linhas. Mesmo com a etapa de vinculação incluída, esse processo é consideravelmente mais rápido.

v Manutenção simplificada

Ao atualizar um programa muito grande, é muito difícil entender exatamente o que está acontecendo. Isto é particularmente verdadeiro se o programador original escreveu em um estilo diferente do seu. Um pedaço menor de código tende a representar uma única função, e é muito mais fácil entender seu funcionamento interno. Portanto, o fluxo lógico torna-se mais óbvio e, quando você faz alterações, é muito menos provável que você introduza efeitos colaterais indesejados.

v Testes simplificados

Unidades de compilação menores incentivam você a testar funções isoladamente. Esse isolamento ajuda a garantir que a cobertura do teste seja concluída; ou seja, que todas as entradas possíveis e caminhos lógicos sejam testados.

v Melhor utilização dos recursos de programação

A modularidade presta-se a uma maior divisão do trabalho. Quando você escreve programas grandes, é difícil (se não impossível) subdividir o trabalho. Codificar todas as partes de um programa pode esticar os talentos de um programador júnior ou desperdiçar as habilidades de um programador sênior. v Migração mais fácil de código de outros sistemas operacionais

**Componentes reutilizáveis**

Com o ILE, você pode selecionar pacotes de rotinas que podem ser misturados em seus próprios programas. Rotinas escritas em qualquer linguagem ILE podem ser usadas por todos os usuários do compilador ILE. O fato de que os programadores podem escrever na linguagem de sua escolha garante que você tenha a mais ampla seleção possível de rotinas.

Os mesmos mecanismos que a IBM e outros fornecedores usam para entregar esses pacotes a você estão disponíveis para uso em seus próprios aplicativos. Sua instalação pode desenvolver seu próprio conjunto de rotinas padrão e fazê-lo em qualquer idioma que escolher.

Não só você pode usar rotinas prontas para uso em suas próprias aplicações. Você também pode desenvolver rotinas na linguagem ILE de sua escolha e comercializá-las para usuários de qualquer linguagem ILE.

**Serviços do Common Runtime**

Uma seleção de componentes prontos para uso (**APIs vinculáveis**) é fornecida como parte do ILE, pronta para ser incorporada em seus aplicativos. Essas APIs fornecem serviços como:

Manipulação de data e hora

Tratamento de mensagens

Rotinas matemáticas

Maior controle sobre o manuseio da tela

Alocação dinâmica de armazenamento

Com o tempo, rotinas adicionais serão adicionadas a esse conjunto e outras estarão disponíveis de fornecedores terceirizados.

A IBM tem informações on-line que fornecem mais detalhes das APIs fornecidas com o ILE. Consulte o tópico APIs que se encontra na categoria Programação do Centro de Informações do IBM i.

**Coexistência com aplicativos existentes**

Os programas ILE podem coexistir com os programas OPM existentes. Os programas ILE podem chamar programas OPM e outros programas ILE. Da mesma forma, os programas OPM podem chamar programas ILE e outros programas OPM. Portanto, com um planejamento cuidadoso, é possível fazer uma transição gradual para o ILE.

**Depurador de origem**

O depurador de origem permite que você depure programas ILE e programas de serviço. Para obter informações sobre o depurador de origem, consulte o Capítulo 10, "Considerações sobre depuração", na página 113.

**Melhor controle sobre os recursos**

Antes da introdução do ILE, os recursos (por exemplo, arquivos abertos) usados por um programa podiam ter como escopo (ou seja, de propriedade de) apenas:

O programa que alocou os recursos O trabalho

Em muitos casos, essa restrição força o designer de aplicativos a fazer compensações.

O ILE oferece uma terceira alternativa. Uma parte do trabalho pode possuir o recurso. Esta alternativa é alcançada através do uso de um construto ILE, o **grupo de ativação**. Em ILE, um recurso pode ter o escopo de qualquer um dos seguintes:

Um programa

Um grupo de ativação O trabalho

#### Caminho de dados abertos compartilhado — cenário

Os caminhos de dados abertos compartilhados (ODPs) são um exemplo de recursos que você pode controlar melhor com o novo nível de escopo do ILE.

Para melhorar o desempenho de um aplicativo, um programador decide usar um ODP compartilhado para o arquivo mestre do cliente. Esse arquivo é usado pelos aplicativos de Entrada de Pedido e de Faturamento.

Como um ODP compartilhado tem o escopo definido para o trabalho, é bem possível que um dos aplicativos inadvertidamente cause problemas para o outro. De fato, evitar tais problemas requer uma coordenação cuidadosa entre os desenvolvedores dos aplicativos. Se os aplicativos foram comprados de diferentes fornecedores, evitar problemas pode até não ser possível.

Que tipo de problemas podem surgir? Considere o seguinte cenário:

1. O arquivo mestre do cliente é codificado no número da conta e contém registros para os números de conta A1, A2, B1, C1, C2, D1, D2 e assim por diante.
2. Um operador está revisando os registros do arquivo mestre, atualizando cada um conforme necessário, antes de solicitar o próximo registro. O registro exibido atualmente é para a conta B1.
3. O telefone toca. O cliente D1 deseja fazer um pedido.
4. O operador pressiona a tecla de função Ir para Entrada de Pedido, processa o pedido para o cliente D1 e retorna à exibição do arquivo mestre.
5. O programa ainda exibe corretamente o registro para B1, mas quando o operador solicita o próximo registro, qual registro é exibido?

Se você disse D2, você está correto. Quando o aplicativo Order Entry lê o registro D1, a posição atual do arquivo foi alterada porque o ODP compartilhado foi definido para o trabalho. Portanto, a solicitação para o próximo registro significa o próximo registro após D1.

Em ILE, esse problema pode ser evitado executando a manutenção do arquivo mestre em um grupo de ativação dedicado ao Faturamento. Da mesma forma, o aplicativo Order Entry seria executado em seu próprio grupo de ativação. Cada aplicativo ainda ganharia os benefícios de um ODP compartilhado, mas cada um teria seu próprio ODP compartilhado, de propriedade do grupo de ativação relevante. Esse nível de escopo evita o tipo de interferência descrito neste exemplo.

O escopo de recursos para um grupo de ativação permite que os programadores tenham a liberdade de desenvolver um aplicativo que seja executado independentemente de quaisquer outros aplicativos em execução no trabalho. Ele também reduz o esforço de coordenação necessário e aprimora a capacidade de gravar extensões drop-in em pacotes de aplicativos existentes.

#### Controle de compromisso — cenário

A capacidade de definir o escopo de um caminho de dados abertos (ODP) compartilhado para o aplicativo é útil na área de controle de compromisso.

Suponha que você deseja usar um arquivo sob controle de compromisso, mas que você também precisa dele para usar um ODP compartilhado. Sem ILE, se um programa abre o arquivo sob controle de compromisso, todos os programas no trabalho têm que fazê-lo. Isso é verdade mesmo que a capacidade de compromisso seja necessária para apenas um ou dois programas.

Um problema potencial com essa situação é que, se algum programa no trabalho emitir uma operação de confirmação, todas as atualizações serão confirmadas. As atualizações são confirmadas mesmo que logicamente não façam parte do aplicativo em questão.

Esses problemas podem ser evitados executando cada parte do aplicativo que requer controle de compromisso em um grupo de ativação separado.

**Melhor controle sobre as interações linguísticas**

Sem o ILE, a maneira como um programa é executado no sistema operacional depende de uma combinação dos seguintes fatores:

O padrão de linguagem (por exemplo, os padrões ANSI para COBOL e C) O desenvolvedor do compilador

Essa combinação pode causar problemas quando você mistura idiomas.

#### Idiomas mistos — cenário

Sem grupos de ativação, que são introduzidos pelo ILE, as interações entre as linguagens OPM são difíceis de prever. Os grupos de ativação de ILE podem resolver essa dificuldade.

Por exemplo, considere os problemas causados pela mistura de COBOL com outras linguagens. O padrão de linguagem COBOL inclui um conceito conhecido como **unidade de execução**. Uma unidade executada agrupa programas de modo que, sob certas circunstâncias, eles se comportem como uma única entidade. Este pode ser um recurso muito útil.

Suponha que três programas CIL COBOL (PRGA, PRGB e PRGC) formem um pequeno aplicativo no qual o PRGA chama PRGB, que por sua vez chama PRGC (consulte a Figura 1). De acordo com as regras do ILE COBOL, esses três programas estão na mesma unidade de execução. Como resultado, se algum deles terminar, todos os três programas devem ser encerrados e o controle deve retornar ao chamador do PRGA.

*Figura 1. Três programas COBOL ILE em uma unidade de execução*

Suponha que agora introduzimos um programa RPG (RPG1) no aplicativo e que RPG1 também é chamado pelo programa COBOL PRGB (consulte a Figura 2 na página 5). Um programa RPG espera que suas variáveis, arquivos e outros recursos permaneçam intactos até que o programa retorne com o indicador de último registro (LR) ativado.

*Figura 2. Três programas ILE COBOL e um programa ILE RPG em uma unidade de execução*

No entanto, o fato de que o programa RPG1 é escrito em RPG não garante que toda a semântica RPG se aplique quando RPG1 é executado como parte da unidade de execução COBOL. Se a unidade de execução terminar, o RPG1 desaparecerá independentemente da configuração do indicador LR. Em muitos casos, essa situação pode ser exatamente o que você quer. No entanto, se o RPG1 é um programa utilitário, talvez controlando a emissão de números de faturas, essa situação é inaceitável.

Podemos evitar essa situação executando o programa RPG em um grupo de ativação separado da unidade de execução COBOL (consulte a Figura 3). Uma unidade de execução COBOL ILE em si é um grupo de ativação.

*Figura*

*3*

*. Programa de RPG ILE em um programa separado*

*Grupo de Ativação*

Para obter informações sobre as diferenças entre uma unidade de execução OPM e uma unidade de execução ILE, consulte o COBOL ILE

Guia do Programador .

**Melhor otimização de código**

O tradutor pode fazer muito mais tipos de otimização para programas ILE do que para programas OPM.

Um compilador habilitado para ILE não produz diretamente um módulo. Primeiro, ele produz uma forma intermediária do módulo e, em seguida, chama o tradutor ILE para traduzir o código intermediário em instruções que podem ser executadas. Usando um código intermediário que é usado como entrada para o tradutor ILE comum, uma otimização adicionada ao tradutor para um idioma ILE pode beneficiar todos os idiomas ILE.

**Qual é a história do ILE?**

O ILE é um estágio na evolução dos modelos de programas IBM i. Cada estágio evoluiu para atender às necessidades em constante mudança dos programadores de aplicativos.

O ambiente de programação fornecido quando o sistema AS/400 foi introduzido pela primeira vez é chamado de modelo de programa original (OPM). No OS/400®  Versão 1 Versão 2, o Extended Program Model (EPM) foi introduzido.

**Descrição do modelo original do programa**

| Os desenvolvedores de aplicativos inserem o código-fonte em um arquivo de origem e compilam esse código-fonte. Se a compilação for | um êxito, um objeto de programa é criado. O conjunto de funções, processos e regras que são usados para | diretamente criar e executar um objeto de programa é conhecido como o *modelo de programa original (OPM).*

Como um compilador OPM gera o objeto de programa, ele gera código adicional. O código adicional inicializa variáveis de programa e fornece qualquer código necessário para processamento especial que é necessário para a linguagem específica. O processamento especial pode incluir o processamento de quaisquer parâmetros de entrada esperados por este programa. Quando um programa deve começar a ser executado, o código adicional gerado pelo compilador torna-se o ponto de partida (ponto de entrada) para o programa.

Um programa é normalmente ativado quando o sistema operacional encontra uma solicitação de chamada. Em tempo de execução, a chamada para outro programa é uma chamada de programa dinâmico. Os recursos necessários para uma chamada de programa dinâmica podem ser significativos. Os desenvolvedores de aplicativos geralmente projetam um aplicativo para consistir em alguns programas grandes que minimizam o número de chamadas dinâmicas de programas.

| A Figura 4 ilustra a relação entre o OPM e o sistema operacional. Como você pode ver, RPG, | COBOL, CL, BASIC e PL/I operam neste modelo. A partir da versão 6.1, o compilador BASIC não é | mais tempo disponível.

A linha quebrada que forma o limite do OPM indica que o OPM é parte integrante do IBM i. Essa integração significa que muitas funções normalmente fornecidas pelo gravador do compilador são incorporadas ao sistema operacional. A padronização resultante das convenções de chamada permite que programas escritos em um idioma chamem livremente aqueles escritos em outro. Por exemplo, um aplicativo escrito em RPG normalmente inclui vários programas CL para emitir substituições de arquivo ou para enviar mensagens.

*Figura*

*4*

*. Relação do OPM com o IBM i*

#### Principais características do OPM

A lista a seguir identifica as principais características do OPM: v Ligação dinâmica

Quando o programa A quer chamar o programa B, ele simplesmente o faz. Esta chamada de programa dinâmico é uma capacidade simples e poderosa. Em tempo de execução, o sistema operacional localiza o programa B e garante que o usuário tenha o direito de usá-lo.

Um programa OPM tem apenas um único ponto de entrada, enquanto que cada procedimento em um programa ILE pode ser um ponto de entrada.

v Compartilhamento limitado de dados

No OPM, um procedimento interno deve compartilhar variáveis com todo o programa, enquanto que, no ILE, cada procedimento pode ter suas próprias variáveis de escopo local.

**Descrição do modelo de programa estendido**

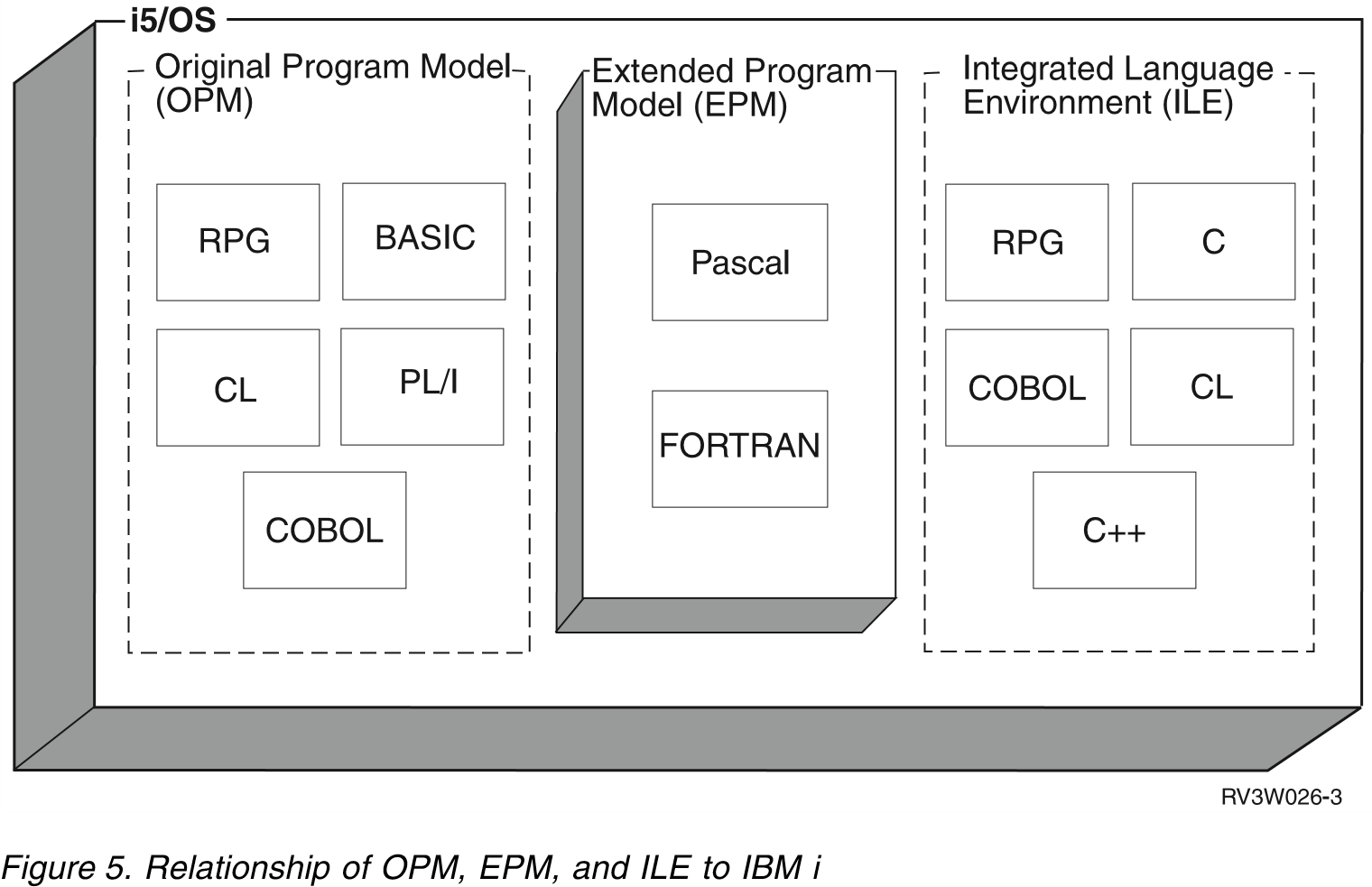
O OPM continua a servir a um propósito útil. No entanto, o OPM não fornece suporte direto para procedimentos conforme definido em linguagens como C. Um *procedimento* é um conjunto de instruções HLL (linguagem de alto nível) autônomas que executa uma tarefa específica e, em seguida, retorna ao chamador. As línguas individuais variam na forma como um procedimento é definido. Em C, um procedimento é chamado de função.

Para permitir que as linguagens que definem chamadas de procedimento entre unidades de compilação ou linguagens que definem procedimentos com variáveis locais sejam executadas no sistema operacional, o OPM foi aprimorado. Esses aprimoramentos são chamados de EPM (Modelo de Programa Estendido). Antes do ILE, o EPM serviu como uma solução provisória para linguagens baseadas em procedimentos, como Pascal e C.

O sistema operativo IBM i já não fornece compiladores EPM.

**Descrição do ambiente linguístico integrado**

Como mostra a Figura 5, o ILE está fortemente integrado ao IBM i, assim como o OPM. Ele fornece o mesmo tipo de suporte para linguagens baseadas em procedimentos que o EPM, mas o faz de forma muito mais completa e consistente. Seu design prevê as linguagens mais tradicionais, como RPG e COBOL, e para futuros desenvolvimentos de linguagem.



#### Principais características das linguagens baseadas em procedimentos

As linguagens baseadas em procedimentos têm as seguintes características:

v Variáveis com escopo local

As variáveis de escopo local são conhecidas apenas dentro do procedimento que as define. O equivalente às variáveis de escopo local é a capacidade de definir duas variáveis com o mesmo nome que se referem a dois dados separados. Por exemplo, a variável COUNT pode ter um comprimento de 4 dígitos no CALCYR sub-rotina e um comprimento de 6 dígitos no CALCDAY sub-rotineiro.

As variáveis de escopo local fornecem benefícios consideráveis quando você escreve sub-rotinas que se destinam a ser copiadas em vários programas diferentes. Sem variáveis de escopo local, os programadores devem usar um esquema como nomear variáveis com base no nome da sub-rotina.

v Variáveis automáticas

As variáveis automáticas são criadas sempre que um procedimento é inserido. As variáveis automáticas são destruídas quando o procedimento é encerrado.

v Variáveis externas

Dados externos são uma maneira de compartilhar dados entre programas. Se o programa A declarar um item de dados como externo, o programa A **exportará** esse item de dados para outros programas que desejam compartilhar esses dados. O programa D pode então **importar** o item sem que os programas B e C estejam envolvidos. Para obter mais informações sobre importações e exportações, consulte "Objeto de módulo" na página 10.

v Vários pontos de entrada

Os programas OPM COBOL e RPG têm apenas um único ponto de entrada. Em um programa COBOL, é o início da DIVISÃO DE PROCEDIMENTOS. Em um programa de RPG, é a saída da primeira página (1P). Este é o modelo que o OPM suporta.

Os idiomas baseados em procedimentos, por outro lado, podem ter vários pontos de entrada. Por exemplo, um programa C pode consistir inteiramente de sub-rotinas a serem usadas por outros programas. Esses procedimentos podem ser exportados, juntamente com dados relevantes, se necessário, para que outros programas sejam importados.

No ILE, programas desse tipo são conhecidos como programas de serviço. Eles podem incluir módulos de qualquer uma das linguagens ILE. Os programas de serviço são semelhantes em conceito às bibliotecas de vínculo dinâmico (DLLs) em

Microsoft Windows. Os programas de serviço são discutidos em maior detalhe em "Programa de Serviço" na página 13.

v Chamadas frequentes

Programas escritos em linguagens baseadas em procedimentos podem ser muito intensivos.

**Capítulo 2. Conceitos básicos do ILE**

A Tabela 1 compara e contrasta o modelo de programa original (OPM) e a Linguagem Integrada

Modelo de ambiente (ILE). Este capítulo explica brevemente as semelhanças e diferenças listadas na tabela.

*Tabela 1. Semelhanças e diferenças entre OPM e ILE*

|  |  |
| --- | --- |
| **OPM** | **ILE** |
| Programa | Programa  Programa de serviço |
| Resultados de compilação em um programa executável | Resultados de compilação em um objeto de módulo não executável |
| Compilar, executar | Compilar, vincular, executar |
| Executar unidades simuladas para cada idioma | Grupos de ativação |
| Chamada dinâmica do programa | Chamada dinâmica do programa  Chamada de procedimento estático |
| Foco em um único idioma | Foco em linguagem mista |
| Tratamento de erros específicos do idioma | Tratamento de erros comuns  Tratamento de erros específicos do idioma |
| Depuradores OPM | Depurador de nível de origem |

**Estrutura de um Programa ILE**

Um programa ILE contém um ou mais módulos. Um módulo, por sua vez, contém um ou mais procedimentos (consulte a Figura 6).

*Figura 6. Estrutura de um Programa ILE*

**Procedimento**

Um *procedimento* é um conjunto de instruções de idioma de alto nível independentes que executa uma tarefa específica e, em seguida, retorna ao chamador. Por exemplo, uma função ILE C é um procedimento ILE.

© Direitos autorais IBM Corp. 1997, 2010

**Objeto Module**

Um **objeto de módulo**  é um *objeto não executável* que é a saída de um compilador ILE. Um objeto de módulo é representado para o sistema pelo símbolo \*MODULE. Um objeto de módulo é o bloco de construção básico para a criação de objetos ILE executáveis. Esta é uma diferença significativa entre ILE e OPM. A saída de um compilador OPM é um programa *executável*.

Um objeto de módulo pode consistir em um ou mais procedimentos e especificações de item de dados. É possível acessar diretamente os procedimentos ou itens de dados em um módulo a partir de outro objeto ILE. Consulte os guias do programador ILE HLL para obter detalhes sobre como codificar os procedimentos e itens de dados que podem ser acessados diretamente por outros objetos ILE.

ILE RPG, ILE COBOL, ILE C e ILE C++ têm os seguintes conceitos comuns:

v Exportações

Uma **exportação** é o nome de um procedimento ou item de dados, codificado em um objeto de módulo, que está disponível para uso por outros objetos ILE. A exportação é identificada por seu nome e seu tipo associado, seja procedimento ou dados.

Uma exportação também pode ser chamada de **definição**. v Importações

Uma **importação** é o uso ou referência ao nome de um procedimento ou item de dados não definido no objeto de módulo atual. A importação é identificada por seu nome e seu tipo associado, seja procedimento ou dados.

Uma importação também pode ser chamada de **referência**.

Um objeto de módulo é o bloco de construção básico de um objeto executável ILE. Portanto, quando um objeto de módulo é criado, o seguinte também pode ser gerado:

v Depurar dados

**Os dados de depuração** são os dados necessários para depurar um objeto ILE em execução. Esses dados são opcionais.

v Procedimento de entrada no programa (PEP)

Um **procedimento**  de entrada de programa é o código gerado pelo compilador que é o ponto de entrada para um programa ILE em uma chamada de programa dinâmico. É semelhante ao código fornecido para o ponto de entrada em um programa OPM. v Procedimento de entrada do usuário (UEP)

Um **procedimento de entrada do usuário**, escrito por um programador, é o alvo da chamada dinâmica do programa. É o procedimento que obtém o controle do PEP. A função main() de um programa C torna-se o UEP desse programa no ILE.

A Figura 7 na página 11 mostra uma exibição conceitual de um objeto de módulo. Neste exemplo, o objeto de módulo M1 exporta dois procedimentos (Draw\_Line e Draw\_Arc) e um item de dados (rtn\_code). O objeto de módulo M1 importa um procedimento chamado Draw\_Plot. Esse objeto de módulo específico tem um PEP, um UEP correspondente (o procedimento Draw\_Arc) e dados de depuração.

**Módulo M1**

|  |
| --- |
| Procedimento de Entrada no Programa (PEP) |
| Procedimento de Entrada do Usuário (UEP): Draw\_Arc |
| |  | | --- | | Draw\_Line Processual;  Dcl rtn\_code EXTRN;  CallPrc Draw\_Plot;  Fim Draw\_Line; | | Draw\_Arc Processual;  Fim Draw\_Arc; | |
| Exportação: Draw\_Line (Procedimento) Draw\_Arc (Procedimento) rtn\_code (Dados) |
| Importação: Draw\_Plot (Procedimento) |
| Depurar dados para o módulo M1 |

RV3W104-0

*Figura 7. Visualização conceitual de um módulo*

Características de um objeto \*MODULE:

v Um objeto \*MODULE é a saída de um compilador ILE. v É o bloco de construção básico para objetos executáveis ILE.

v Não é um objeto executável. v Pode ter um PEP definido.

v Se um PEP é definido, um UEP também é definido. v Pode exportar nomes de procedimentos e itens de dados. v Pode importar nomes de procedimentos e itens de dados. v Pode ter dados de depuração definidos.

**Programa ILE**

Um programa ILE compartilha as seguintes características com um programa OPM: v O programa obtém controle por meio de uma chamada dinâmica do programa.

v Há apenas um ponto de entrada para o programa. v O programa é identificado para o sistema pelo símbolo \*PGM.

Um programa ILE tem as seguintes características que um programa OPM não tem: v Um programa ILE é criado a partir de um ou mais objetos de módulo copiados. v Um ou mais dos módulos copiados podem conter um PEP.

v Você tem controle sobre qual PEP do módulo é usado como o PEP para o objeto de programa ILE.

Quando o comando Create Program (CRTPGM) é especificado, o parâmetro ENTMOD permite que você selecione qual módulo contendo um PEP é o ponto de entrada do programa.

Um PEP que está associado a um módulo que não está selecionado como o ponto de entrada para o programa é ignorado. Todos os outros procedimentos e itens de dados do módulo são usados conforme especificado. Apenas o PEP é ignorado.

Quando uma chamada de programa dinâmico é feita para um programa ILE, o PEP do módulo que foi selecionado no momento da criação do programa recebe controle. O PEP chama o UEP associado.

Quando um objeto de programa ILE é criado, somente os procedimentos associados aos módulos copiados que contêm dados de depuração podem ser depurados pelo depurador ILE. Os dados de depuração não afetam o desempenho de um programa ILE em execução.

A Figura 8 mostra uma exibição conceitual de um objeto de programa ILE. Quando o programa PGMEXAMP é chamado, o PEP do programa, que foi definido no objeto de módulo copiado M3, recebe controle. O módulo copiado M2 também tem um PEP definido, mas é ignorado e nunca usado pelo programa.

Neste exemplo de programa, apenas dois módulos, M1 e M3, têm os dados necessários para o depurador ILE. Os procedimentos dos módulos M2 e M4 não podem ser depurados com o depurador ILE.

Os procedimentos importados print e SIN são resolvidos para procedimentos exportados dos programas de serviço PRINTS e MATHFUNC, respectivamente.

**\*PGM (PGMEXAMP)**

Procedimento P3;

Fim P3;

Procedimento P1;

DCL D EXTRN;

Impressão CallPrc;

Fim P1;

Procedimento P4;

DCL X REAL;

D=SIN(X);

Fim P4;

Procedimento P2;

Fim P2;

RV2W980-5

Procedimento de Entrada no Programa (Use PEP no módulo M3)

Procedimento de entrada do usuário: (Use P3 no módulo M3)

Importações resolvidas internamente: P1, P2, P4, D

PEP usado: Definido no módulo M3

UEP: Procedimento P3 no módulo M3

Importações resolvidas externamente:

imprimir em \*LIBL/PRINTS

PECADO em MATHLIB/MATHFUNC

Depurar dados

Depurar dados

PEP

UEP: P3

PEP

UEP: P2

CallPrc P2;

CallPrc P1;

CallPrc P4;

Módulo M3

Módulo M1

Módulo M4

Módulo M2

*Figura 8. Visão conceitual de um programa ILE*

Características de um objeto ILE \*PGM:

v Um ou mais módulos de qualquer linguagem ILE são copiados para criar o objeto \*PGM.

v A pessoa que cria o programa tem controle sobre qual PEP do módulo se torna o único PEP para o programa.

v Em uma chamada de programa dinâmico, o PEP do módulo que foi selecionado como o PEP para o programa obtém o controle para ser executado.

v O UEP associado ao PEP selecionado é o ponto de entrada do usuário para o programa. v Procedimentos e nomes de itens de dados não podem ser exportados do programa.

v Procedimentos ou nomes de itens de dados podem ser importados de módulos e programas de serviço, mas não de objetos de programa. Para obter informações sobre programas de serviço, consulte "Programa de serviço".

v Os módulos podem ter dados de depuração. v Um programa é um objeto executável.

**Programa de Serviço**

Um **programa**  de serviço é uma coleção de procedimentos executáveis e itens de dados disponíveis fácil e diretamente acessíveis por outros programas ILE ou programas de serviço. Em muitos aspectos, um programa de serviço é semelhante a uma biblioteca de sub-rotina ou biblioteca de procedimentos.

Os programas de serviço fornecem serviços comuns que outros objetos ILE podem precisar; daí o programa de serviço de nome. Um exemplo de um conjunto de programas de serviço fornecidos pelo sistema operacional são os procedimentos de tempo de execução para uma linguagem. Esses procedimentos de tempo de execução geralmente incluem itens como procedimentos matemáticos e procedimentos comuns de entrada e saída.

A **interface pública** de um programa de serviço consiste nos nomes dos procedimentos exportados e itens de dados acessíveis por outros objetos ILE. Somente os itens que são exportados dos objetos de módulo que compõem um programa de serviço são elegíveis para serem exportados de um programa de serviço.

O programador pode especificar quais procedimentos ou itens de dados podem ser conhecidos por outros objetos ILE. Portanto, um programa de serviço pode ter procedimentos e dados ocultos ou privados que não estão disponíveis para nenhum outro objeto ILE.

É possível atualizar um programa de serviço sem ter que recriar os outros programas ILE ou programas de serviço que usam o programa de serviço atualizado. O programador que faz as alterações no programa de serviço controla se a alteração é compatível com o suporte existente.

A maneira como o ILE fornece para você controlar alterações compatíveis é usando a **linguagem do fichário**. O idioma do fichário permite que você defina a lista de nomes de procedimento e nomes de item de dados que podem ser exportados. Uma **assinatura** é gerada a partir dos nomes de procedimentos e itens de dados e da ordem em que eles são especificados no idioma do fichário. Para fazer alterações compatíveis em um programa de serviço, novos nomes de procedimento ou item de dados devem ser adicionados ao final da lista de exportação. Para obter mais informações sobre assinaturas, a linguagem do fichário e a proteção do investimento de seus clientes em seus programas de serviço, consulte "Linguagem do fichário" na página 67.

A Figura 9 na página 14 mostra uma exibição conceitual de um programa de serviço. Observe que os módulos que compõem esse programa de serviço são o mesmo conjunto de módulos que compõem o objeto de programa ILE PGMEXAMP na Figura 8 na página 12. A assinatura anterior, Sigyy, para o programa de serviço SPGMEXAMP contém os nomes dos procedimentos P3 e P4. Depois que uma alteração compatível com o aumento é feita no programa de serviço, a assinatura atual, Sigxx, contém não apenas os nomes dos procedimentos P3 e P4; ele também contém o nome do item de dados D. Outros programas ILE ou programas de serviço que usam procedimentos P3 ou P4 não precisam ser recriados.

| Embora os módulos em um programa de serviço possam ter PEPs, esses PEPs são ignorados. O programa de serviço | em si não tem um PEP. Portanto, ao contrário de um objeto de programa, um programa de serviço não pode ser o destino de um | chamada dinâmica do programa.

P3

P4

D

**Público**

**Interface**

**\*SRVPGM**

**(**

**SPGMEXAMP**

**)**

Procedimento P3;

Fim P3;

Procedimento P1;

DCL D EXTRN;

Fim P1;

Procedimento P4;

DCL X REAL;

Fim P4;

Procedimento P2;

Fim P2;

RV2W981-8

Importações resolvidas internamente: P1, P2, P4, D

Assinatura atual = Sigxx

Assinatura Anterior = Sigyy

Importações resolvidas externamente:

Depurar dados

Depurar dados

PEP

UEP: A3

PEP

UEP: A2

CallPrc P2;

Impressão CallPrc;

D=SIN(X);

CallPrc P1;

CallPrc P4;

Módulo M3

Módulo M1

Módulo M4

Módulo M2

imprimir em \*LIBL/PRINTS

PECADO em MATHLIB/MATHFUNC

*Figura 9. Exibição conceitual de um programa de serviço ILE*

Características de um objeto ILE \*SRVPGM:

v Um ou mais módulos de qualquer linguagem ILE são copiados para criar o objeto \*SRVPGM.

v Nenhum PEP está associado ao programa de serviço. Como não há PEP, uma chamada de programa dinâmico para um programa de serviço não é válida. O PEP de um módulo é ignorado.

v Outros programas ILE ou programas de serviço podem usar as exportações deste programa de serviço identificadas pela interface pública.

v As assinaturas são geradas a partir do procedimento e dos nomes de item de dados que são exportados do programa de serviço.

v Os programas de serviço podem ser substituídos sem afetar os programas ILE ou os programas de serviço que os utilizam, desde que as assinaturas anteriores ainda sejam suportadas.

v Os módulos podem ter dados de depuração.

| v Um programa de serviço é uma coleção de itens de dados e procedimentos executáveis.

v Dados fracos podem ser exportados somente para um grupo de ativação. Ele não pode ser feito parte da interface pública que é exportada do programa de serviço. Para obter informações sobre dados fracos, consulte Exportar em "Conceitos de importação e exportação" na página 66.

**Diretório de Vinculação**

| Um **diretório de associação** lista os nomes de módulos e programas de serviço que você pode precisar ao criar

| um programa ILE ou programa de serviço. Módulos ou programas de serviço listados em um diretório de vinculação são usados | somente se eles fornecerem uma exportação que possa satisfazer quaisquer solicitações de importação atualmente não resolvidas. Um | de ligação diretório é um objeto do sistema que é identificado para o sistema pelo símbolo \*BNDDIR.

Os diretórios de vinculação são opcionais. As razões para usar diretórios de vinculação são conveniência e tamanho do programa.

| v Eles oferecem um método conveniente de listar os módulos ou programas de serviço que você pode precisar quando

| criando seu próprio programa ILE ou programa de serviço. Por exemplo, um diretório de vinculação pode listar todos os | módulos e programas de serviço que fornecem funções matemáticas. Se você quiser usar alguns desses | , você especifica apenas um diretório de vinculação, não cada módulo ou programa de serviço que você usa.

| **Nota:** Quanto mais módulos ou programas de serviço um diretório de vinculação lista, mais tempo pode levar para vincular | os programas. Portanto, você deve incluir apenas os módulos ou programas de serviço necessários em | seu diretório de vinculação.

v Diretórios de vinculação podem reduzir o tamanho do programa porque você não especifica módulos ou programas de serviço que não são usados.

Muito poucas restrições são colocadas nas entradas em um diretório de vinculação. O nome de um módulo ou programa de serviço pode ser adicionado a um diretório de vinculação mesmo que esse objeto ainda não exista.

Para obter uma lista de comandos CL usados com diretórios de vinculação, consulte o Apêndice C, "Comandos CL usados com objetos ILE", na página 183.

A Figura 10 mostra uma exibição conceitual de um diretório de vinculação.

**Diretório de Vinculação (ABD)**

|  |  |  |
| --- | --- | --- |
| Nome do objeto Tipo de objeto | | Biblioteca de objetos |
| QALLOC QMATH  QFREEQHFREE.  .  . | \*SRVPGM  \*SRVPGM  \*MÓDULO\*SRVPGM.  .  . | \*LIBL  QSYS  \*LIBLABC.  .  . |

RV2W982-0 *Figura 10. Exibição conceitual de um diretório de vinculação*

Características de um objeto \*BNDDIR:

v Método conveniente de agrupar os nomes de programas de serviço e módulos que podem ser necessários para criar um programa ILE ou programa de serviço.

v Como as entradas de diretório de vinculação são apenas nomes, os objetos listados ainda não precisam existir no sistema.

v Os únicos nomes de biblioteca válidos são \*LIBL ou uma biblioteca específica.

v Os objetos na lista são opcionais. Os objetos nomeados serão usados somente se existirem importações não resolvidas e se o objeto nomeado fornecer uma exportação para satisfazer a solicitação de importação não resolvida.

**Vinculação de processamento de diretório**

Durante a vinculação, o processamento ocorre nesta ordem:

1. Todos os módulos especificados no parâmetro MODULE são examinados. O fichário determina a lista de símbolos importados e exportados pelo objeto. Depois de examinados, os módulos são vinculados, na ordem listada, ao programa que está sendo criado.
2. Todos os programas de serviço no parâmetro BNDSRVPGM são examinados na ordem listada. Os programas de serviço são vinculados somente se necessário para resolver uma importação.
3. Todos os diretórios de vinculação no parâmetro BNDDIR são processados na ordem listada. Todos os objetos listados nesses diretórios de vinculação são examinados na ordem listada, mas são vinculados somente se necessário para resolver importações. Entradas duplicadas em diretórios de vinculação são silenciosamente ignoradas.
4. Cada módulo tem uma lista de **objetos do sistema de referência**. Esta lista é simplesmente uma lista de diretórios de ligação. Os objetos do sistema de referência dos módulos acoplados são processados de modo que todos os objetos do sistema de referência do primeiro módulo sejam processados primeiro, depois os objetos do segundo módulo e assim por diante. Os objetos listados nesses diretórios de vinculação são examinados na ordem listada, somente conforme necessário, e vinculados somente se necessário. Esse processamento continua apenas enquanto existirem importações não resolvidas, mesmo que OPTION(\*UNRSLVREF) seja usado. Em outras palavras, o processamento de objetos pára quando todas as importações são resolvidas.

Enquanto os objetos são examinados, a mensagem CPD5D03, "Definição fornecida várias vezes para o símbolo", pode ser sinalizada mesmo que o objeto não esteja finalmente vinculado ao programa que está sendo criado.

Observe que os módulos geralmente têm importações que não são aparentes do código-fonte do módulo. Eles são adicionados pelo compilador para implementar vários recursos de linguagem que exigem suporte de tempo de execução de programas de serviço. Use DSPMOD DETAIL(\*IMPORT) para ver essas importações.

Para ver a lista de símbolos importados e exportados para um módulo ou programa de serviço, consulte a seção Listagem de Informações do Fichário de uma listagem CRTPGM ou CRTSRVPGM DETAIL(\*EXTENDED). Ele lista os objetos que são examinados durante a associação.

Os objetos de módulo ou programa de serviço que estão vinculados ao programa ou programa de serviço que está sendo criado são indicados na seção Listagem de Informações do Fichário de uma listagem CRTPGM ou CRTSRVPGM DETAIL(\*EXTENDED). Depois que um objeto é criado, você também pode usar o comando DSPPGM ou DSPSRVPGM

DETAIL(\*MODULE) para ver os objetos \*MODULE acoplados e DETAIL(\*SRVPGM) para ver a lista de objetos \*SRVPGM acoplados.

| Você pode usar DSPMOD DETAIL(\*REFSYSOBJ) para ver a lista de objetos do sistema de referência, que são de ligação

| Diretórios. Esses diretórios de associação normalmente listam os nomes das APIs do programa de serviço fornecidas pelo

| suporte ao sistema operacional ou ao tempo de execução da linguagem. Desta forma, um módulo pode ser vinculado à sua linguagem | suporte de tempo de execução e APIs do sistema sem que o programador tenha que especificar nada de especial no

| comando.

**Funções do fichário**

A função do fichário é semelhante, mas um pouco diferente da função fornecida por um editor de ligação. O **fichário** processa solicitações de importação para nomes de procedimento e nomes de item de dados de módulos especificados. Em seguida, o fichário tenta localizar as exportações correspondentes nos módulos, programas de serviço e diretórios de vinculação especificados.

Ao criar um programa ILE ou um programa de serviço, o fichário executa os seguintes tipos de associação:

v Vincular por cópia

Para criar o programa ILE ou programa de serviço, os seguintes itens são copiados:

Os módulos especificados no parâmetro module

Quaisquer módulos selecionados no diretório de associação que forneçam uma exportação para uma importação não resolvida

Os endereços físicos dos procedimentos necessários e os itens de dados usados nos módulos copiados são estabelecidos quando o programa ILE ou o programa de serviço é criado.

Por exemplo, na Figura 9 da página 14, o procedimento P3 no módulo M3 chama o procedimento P2 no módulo M2. O endereço físico do procedimento P2 no módulo M2 é dado a conhecer ao procedimento M3 para que o endereço possa ser acessado diretamente.

v Vincular por referência

Links simbólicos para os programas de serviço que fornecem exportações para solicitações de importação não resolvidas são salvos no programa criado ou programa de serviço. Os links simbólicos referem-se aos programas de serviço que fornecem as exportações. Os links são convertidos em endereços físicos quando o objeto de programa ao qual o programa de serviço está vinculado é ativado.

A Figura 9 na página 14 mostra um exemplo de um link simbólico para o SIN no programa de serviço

\*MATHLIB/MATHFUNC. O link simbólico para SIN é convertido em um endereço físico quando o objeto de programa ao qual o programa de serviço SPGMEXAMP está vinculado é ativado.

Em tempo de execução, com links físicos estabelecidos para os procedimentos e itens de dados que estão sendo usados, há pouca diferença de desempenho entre o seguinte: v Acessando um procedimento local ou item de dados v Acessando um procedimento ou item de dados em um módulo ou programa de serviço diferente vinculado ao mesmo

programa

A Figura 11 e a Figura 12 na página 18 mostram visões conceituais de como o programa ILE PGMEXAMP e o programa de serviço SPGMEXAMP foram criados. O fichário usa os módulos M1, M2, M3 e M4 e os programas de serviço PRINTS e MATHFUNC para criar o programa ILE PGMEXAMP e o programa de serviço SPGMEXAMP.

Módulo M1 Módulo M2 Módulo M3 Módulo M4 PRINTS MATHFUNC

**Programas de Serviço**

CRTPGM PGM (PGMEXAMP) MÓDULO (M1, M2, M3, M4) ENTMOD(\*LIBL/M3) +

BNDSRVPGM(\*LIBL/PRINTS MATHLIB/MATHFUNC)

Fichário

Programa

PGMEXAMP

RV2W983-3

*Figura 11. Criação de um Programa ILE.* A linha quebrada indica que os programas de serviço estão vinculados por referência em vez de serem vinculados por cópia.

Módulo M1 Módulo M2 Módulo M3 Módulo M4 PRINTS MATHFUNC

**Programas de Serviço**

CRTSRVPGM SRVPGM (SPGMEXAMP) MÓDULO (M1, M2, M3, M4) EXPORTAÇÃO (\*SRCFILE) +

SRCFILE(\*LIBL/QSRVSRC) SRCMBR(\*SRVPGM) BNDSRVPGM(\*LIBL/PRINTS MATHLIB/MATHFUNC)

Fichário

Programa de Serviço

SPGMEXAMP

RV3W030-1

*Figura 12. Criação de um Programa de Serviços.* A linha quebrada indica que os programas de serviço estão vinculados por referência em vez de serem vinculados por cópia.

Para obter informações adicionais sobre como criar um programa ILE ou um programa de serviço, consulte o Capítulo 5, "Conceitos de criação de programas", na página 55.

**Chamando um programa ou um procedimento**

No ILE você pode chamar um programa ou um procedimento. O ILE requer que o chamador identifique se o destino da instrução de chamada é um programa ou um procedimento. As linguagens ILE comunicam esse requisito com instruções de chamada separadas para programas e procedimentos. Portanto, quando você escreve seu programa ILE, você deve saber se você está chamando um programa ou um procedimento.

Cada linguagem ILE tem uma sintaxe exclusiva que permite distinguir entre uma chamada de programa dinâmico e uma chamada de procedimento estático. A instrução de chamada padrão em cada idioma ILE tem como padrão uma chamada de programa dinâmico ou uma chamada de procedimento estático. Para RPG e COBOL o padrão é uma chamada de programa dinâmico, e para C o padrão é uma chamada de procedimento estático. Assim, a chamada de linguagem padrão executa o mesmo tipo de função no OPM ou no ILE. Essa convenção torna a migração de uma linguagem OPM para uma linguagem ILE relativamente fácil.

Para determinar quanto tempo os nomes do seu procedimento podem ter, consulte o guia do programador ILE HLL.

**Chamadas dinâmicas de programas**

Uma chamada de programa dinâmico transfere controle para um objeto de programa ILE ou um objeto de programa OPM, mas não para um programa de serviço ILE. As chamadas dinâmicas do programa incluem o seguinte:

v Um programa OPM pode chamar outro programa OPM ou um programa ILE v Um programa ILE pode chamar um programa OPM ou outro programa ILE v Um programa de serviço pode chamar um programa OPM ou um programa ILE

**Chamadas de procedimento estático**

Uma chamada de procedimento estático transfere o controle para um procedimento ILE. As chamadas de procedimento estático podem ser codificadas somente em idiomas ILE. Uma chamada de procedimento estático pode ser usada para chamar qualquer um dos seguintes:

v Um procedimento dentro do mesmo módulo v Um procedimento em um módulo separado dentro do mesmo programa ILE ou programa de serviço v Um procedimento em um programa de serviço ILE separado

A Figura 13 mostra exemplos de chamadas de procedimento estático. A figura mostra que:

v Um procedimento em um programa ILE pode chamar um procedimento exportado no mesmo programa ou em um programa de serviço. O procedimento P1 no programa A chama o procedimento P2 em outro módulo copiado. Procedimento P3 no programa C chama o procedimento P4 no programa de serviço D.

v Um procedimento em um programa de serviço pode chamar um procedimento exportado no mesmo programa de serviço ou em outro programa de serviço. O procedimento P6 no programa de serviço B chama o procedimento P7 em outro módulo copiado. Procedimento P5 no programa de serviço E chama o procedimento P4 no programa de serviço F.

**Programa A**

**Programa C**

Chamada de procedimento estático

Estático |

Procedimento

Chamar

Estático |

Procedimento

Chamar

Chamada de procedimento estático

**Programa de Serviço E**

**Programa de Serviço B**

**Programa de Serviço D**

**Programa de Serviço F**

Módulo |

Módulo |

Módulo |

Módulo |

Módulo |

Módulo |

Módulo |

Módulo |

Proc: P1

CallPrc P2

Fim P1

Proc: P3

CallPrc P4

Fim P3

Proc: P5

CallPrc P4

Fim P5

Proc: P6

CallPrc P7

Fim P6

Proc: P4

Fim P4

Proc: P4

Fim P4

Proc: P2

Fim P2

Proc: P7

Fim P7

RV2W993-2 *Figura 13. Chamadas de procedimento estático*

| **Chamadas de ponteiro de procedimento**

| Consulte o Capítulo 7, "Chamadas para procedimentos e programas", na página 91 para obter informações sobre o ponteiro do procedimento | Chamadas.

**Ativação**

Depois de criar com êxito um programa ILE, você desejará executar seu código. O processo de obter um programa ou programa de serviço pronto para ser executado é chamado *de ativação*. Não é necessário emitir um comando para ativar um programa. A ativação é feita pelo sistema quando um programa é chamado.

| A ativação executa as seguintes funções:

| v Aloca e inicializa os dados estáticos necessários para o programa ou programa de serviço

| v Resolve as importações para endereços de tempo de execução das exportações do programa de serviço correspondente.

| Um programa ou programa de serviço pode ser ativado em mais de um grupo de ativação, mesmo dentro do mesmo | trabalho. Cada ativação é local para um grupo de ativação específico e cada ativação tem seu próprio armazenamento estático. | Quando um programa ou programa de serviço é usado simultaneamente por muitos trabalhos, apenas uma cópia do | as instruções residem no armazenamento, mas as variáveis estáticas são separadas para cada ativação.

Por padrão, os programas de serviço são imediatamente ativados durante a chamada para um programa que, direta ou indiretamente, requer seus serviços. Você pode solicitar a ativação adiada para um programa de serviço quando estiver vinculando um programa ILE ou um programa de serviço criado para V6R1 ou posterior. Se você solicitar a ativação adiada para um programa de serviço, a ativação de um programa de serviço poderá ser adiada até que um de seus procedimentos importados seja chamado. Para minimizar os custos de ativação na inicialização do programa e durante a execução do programa, sugere-se que você especifique a ativação adiada para os programas de serviço que satisfazem as importações de procedimento e que são usados somente em caminhos de código percorridos com pouca frequência.

**Anotações:**

1. Se você solicitar a ativação adiada para um programa de serviço que satisfaça uma importação de dados, a ativação imediata parcial será necessária para inicializar os dados estáticos.
2. Se você solicitar a ativação adiada para um programa de serviço que satisfaça uma importação de procedimento para uma chamada de ponteiro de procedimento, a ativação imediata parcial será necessária para fornecer a associação para a chamada de ponteiro de procedimento.

Para especificar o modo de ativação de um programa de serviço como imediato ou adiado, use \*IMMED ou \*DEFER no parâmetro BNDSRVPGM dos seguintes comandos CL:

v Criar Programa (CRTPGM) v Criar Programa de Serviço (CRTSRVPGM) v Programa de Atualização (UPDPGM) v Programa de Serviço de Atualização (UPDSRVPGM)

O comando Add Binding Directory (ADDBNDDIRE) fornece um campo de entrada semelhante para uma entrada de programa de serviço e o comando Work with Binding Directory Entries (WRKBNDDIRE) fornece a saída do modo de ativação de entradas de programa de serviço em um diretório de associação.

Se uma das seguintes opções for verdadeira:

v A ativação não pode localizar o programa de serviço necessário v O programa de serviço não oferece mais suporte aos procedimentos ou itens de dados representados pela assinatura ocorre um erro e você não pode executar seu aplicativo.

Para obter mais detalhes sobre a ativação do programa, consulte "Criação de ativação do programa" na página 24.

Quando a ativação aloca o armazenamento necessário para as variáveis estáticas usadas por um programa, o espaço é alocado de um grupo de ativação. No momento em que o programa ou programa de serviço é criado, você pode especificar o grupo de ativação que deve ser usado em tempo de execução.

Para obter mais informações sobre grupos de ativação, consulte "Grupo de ativação" na página 24.

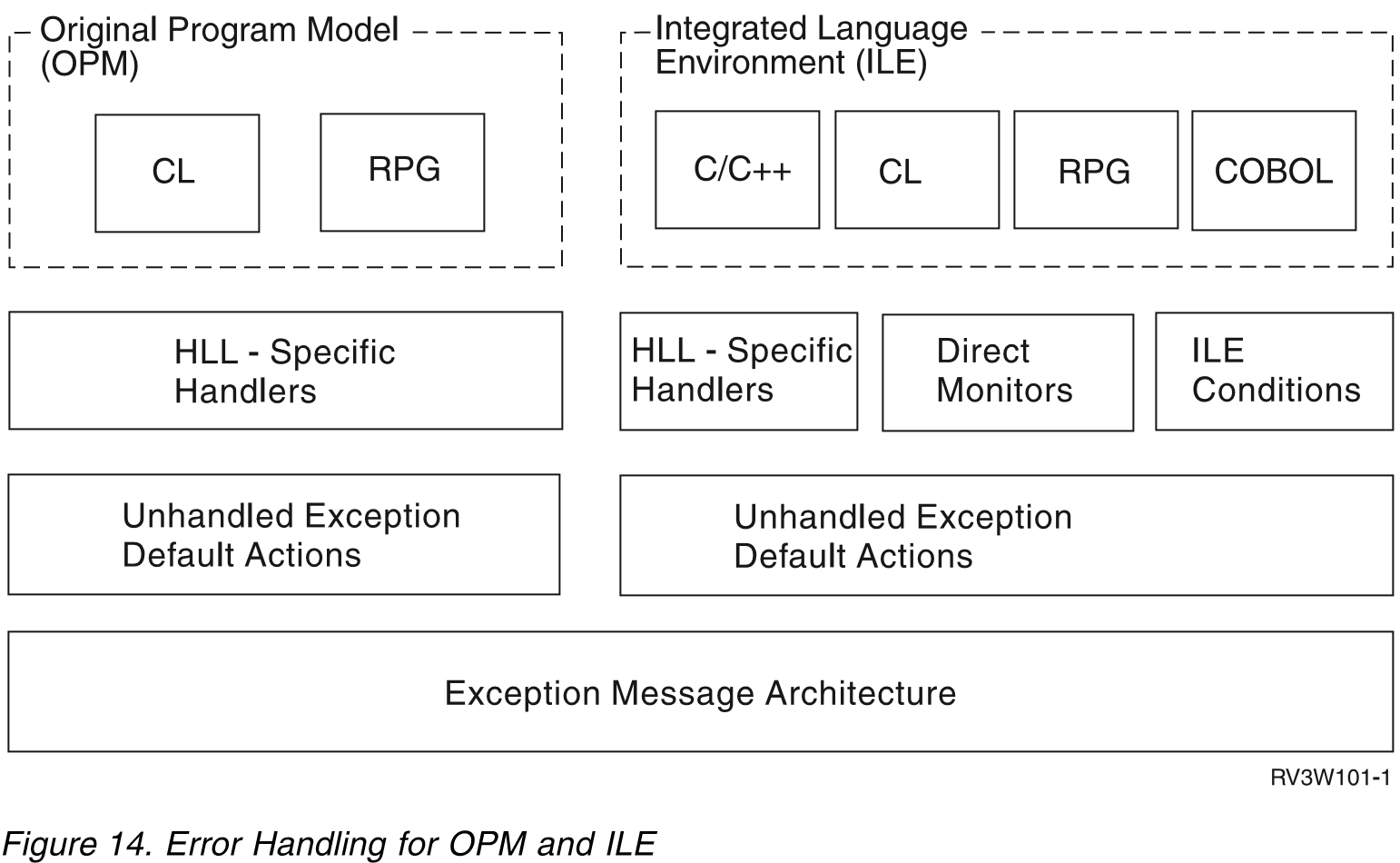
**Visão geral do tratamento de erros**

A Figura 14 mostra a estrutura completa de tratamento de erros para os programas OPM e ILE. Esta figura é usada em todo este manual para descrever os recursos avançados de tratamento de erros. Este tópico fornece uma breve visão geral dos recursos de tratamento de erros de linguagem padrão. Para obter informações adicionais sobre o tratamento de erros, consulte "Tratamento de erros" na página 34.

A figura mostra uma camada fundamental chamada arquitetura de mensagem de exceção. Uma mensagem de exceção pode ser gerada pelo sistema sempre que um programa OPM ou um programa ILE encontrar um erro. Mensagens de exceção também são usadas para comunicar informações de status que podem não ser consideradas um erro de programa. Por exemplo, uma condição de que um registro de banco de dados não é encontrado é comunicada enviando uma mensagem de exceção de status.

Cada linguagem de alto nível define os recursos de tratamento de erros específicos do idioma. Embora esses recursos variem de acordo com o idioma, em geral, é possível que cada usuário HLL declare a intenção de lidar com situações de erro específicas. A declaração dessa intenção inclui a identificação de uma rotina de tratamento de erros. Quando ocorre uma exceção, o sistema localiza a rotina de tratamento de erros e passa o controle para instruções escritas pelo usuário. Você pode executar várias ações, incluindo encerrar o programa ou se recuperar do erro e continuar.

A Figura 14 mostra que o ILE usa a mesma arquitetura de mensagem de exceção usada pelos programas OPM. As mensagens de exceção geradas pelo sistema iniciam o tratamento de erros específicos do idioma em um programa ILE, assim como fazem em um programa OPM. A camada mais baixa da figura inclui a capacidade de enviar e receber mensagens de exceção. Isso pode ser feito com APIs ou comandos do manipulador de mensagens. Mensagens de exceção podem ser enviadas e recebidas entre programas ILE e OPM.



O tratamento de erros específicos do idioma funciona de forma semelhante para programas ILE e para programas OPM, mas há diferenças básicas:

v Quando o sistema envia uma mensagem de exceção para um programa ILE, o procedimento e o nome do módulo são usados para qualificar a mensagem de exceção. Se você enviar uma mensagem de exceção, essas mesmas qualificações poderão ser especificadas. Quando uma mensagem de exceção aparece no log de trabalho de um programa ILE, o sistema normalmente fornece o nome do programa, o nome do módulo e o nome do procedimento.

v A otimização extensiva para programas ILE pode resultar em vários números de instrução HLL associados às mesmas instruções geradas. Como resultado da otimização, as mensagens de exceção que aparecem no log de trabalho podem conter vários números de instrução HLL.

Recursos adicionais de tratamento de erros são descritos em "Tratamento de Erros" na página 34.

**Otimizando o Tradutor**

*Otimização* significa maximizar o desempenho do tempo de execução do objeto. Todas as linguagens ILE têm acesso às técnicas de otimização fornecidas pelo tradutor otimizador ILE. Geralmente, quanto maior o nível de otimização, mais tempo leva para criar o objeto. Em tempo de execução, programas altamente otimizados ou programas de serviço devem ser executados mais rapidamente do que os programas correspondentes ou programas de serviço criados com um nível mais baixo de otimização.

Embora a otimização possa ser especificada para um módulo, objeto de programa e programa de serviço, as técnicas de otimização se aplicam a módulos individuais. Os níveis de otimização são:

10 ou \*NENHUM

20 ou \*BÁSICO

30 ou \*COMPLETO

40 (mais otimização do que o nível 30)

Por razões de desempenho, é provavelmente desejável usar um alto nível de otimização quando um programa é colocado em produção. Para testes iniciais, pode ser necessário usar um nível de otimização mais baixo devido a limitações de depuração. No entanto, é altamente recomendável que você use o nível de otimização no qual um programa é liberado para teste final porque alguns bugs, como dados não inicializados, podem ser expostos apenas em níveis de otimização mais altos.

Como a otimização no nível 30 (\*FULL) ou no nível 40 pode afetar significativamente as instruções do programa, talvez seja necessário estar ciente de certas limitações de depuração e detecção de exceção de endereçamento diferente. Consulte o Capítulo 10, "Considerações sobre depuração", na página 113 para obter considerações sobre depuração. Consulte o Apêndice B, "Exceções em programas otimizados", na página 181 para abordar considerações de erro.

**Depurador**

O ILE fornece um depurador que permite a depuração no nível da origem. O depurador pode trabalhar com um arquivo de listagem e permitir que você defina pontos de interrupção, exiba variáveis e entre ou sobre uma instrução. Você pode fazer isso sem precisar inserir um comando a partir da linha de comando. Uma linha de comando também está disponível ao trabalhar com o depurador.

O depurador de nível de origem usa APIs fornecidas pelo sistema para permitir que você depure seu programa ou programa de serviço. Essas APIs estão disponíveis para todos e permitem que você escreva seu próprio depurador.

Os depuradores para programas OPM continuam a existir no sistema operacional, mas podem ser usados para depurar apenas programas OPM. No entanto, o depurador ILE pode depurar programas OPM que são compilados com OPTION(\*SRCDBG) ou OPTION(\*LSTDBG).

Depurar um programa otimizado pode ser difícil. Quando você usa o depurador ILE para exibir ou alterar uma variável que é usada por um programa ou procedimento em execução, o depurador recupera ou atualiza os dados no local de armazenamento para essa variável. No nível 20 (\*BASIC), 30 (\*FULL) ou 40 otimização, o valor atual de uma variável de dados pode não estar em armazenamento, portanto, o depurador não pode acessá-lo. Assim, o valor exibido para uma variável pode não ser o valor atual. Por esse motivo, você deve usar o nível de otimização 10 (\*NONE) para criar módulos durante o desenvolvimento. Em seguida, para obter o melhor desempenho, você deve usar o nível de otimização 30 (\*FULL) ou 40 ao criar módulos para teste final antes que um programa seja colocado em produção.

Para obter mais informações sobre o depurador ILE, consulte o Capítulo 10, "Considerações sobre depuração", na página 113.

**Capítulo 3. Conceitos avançados de ILE**

Este capítulo descreve conceitos avançados para o modelo ILE. Antes de ler este capítulo, você deve estar familiarizado com os conceitos descritos no Capítulo 2, "Conceitos Básicos da ILE", na página 9.

**Ativação do programa**

| **A ativação** é o processo usado para preparar um programa a ser executado. Os programas ILE e os programas de serviço | devem ser ativados pelo sistema antes que possam ser executados.

A ativação do programa inclui duas etapas principais:

1. Aloque e inicialize o armazenamento estático para o programa.
2. Conclua a associação de programas a programas de serviço.

Este tópico se concentra na etapa 1. A etapa 2 é explicada em "Ativação do programa de serviço" na página 30.

A Figura 15 mostra um objeto de programa armazenado no armazenamento em disco permanente. Tal como acontece com todos os objetos do IBM i, os objetos de programa podem ser partilhados por vários utilizadores simultâneos em execução em tarefas diferentes e existe apenas uma cópia das instruções do programa. No entanto, quando um programa é ativado, o armazenamento para variáveis de programa deve ser alocado e inicializado.

Como mostra a Figura 15, cada ativação de programa tem sua própria cópia dessas variáveis.

Programa A

Programa

Instruções

Uma cópia do

instruções do programa

**Trabalho**

**Trabalho**

Grupo de Ativação

Grupo de Ativação

Programa A

Programa A

Variável X = 10

Variável X = 20

Uma cópia de variáveis estáticas

para cada ativação do programa

RV2W986-3

*Figura 15. Uma cópia de variáveis estáticas para cada ativação do programa*

© Direitos autorais IBM Corp. 1997, 2010

**Criação de ativação do programa**

O ILE gerencia o processo de ativação do programa acompanhando as ativações do programa dentro de um grupo de ativação. Consulte "Grupo de ativação" para obter uma definição de um grupo de ativação. Apenas uma ativação para um objeto de programa específico está em um grupo de ativação. Programas de mesmo nome que residem em bibliotecas diferentes são considerados objetos de programa diferentes quando essa regra é aplicada.

Quando você usa uma instrução de chamada de programa dinâmico em seu programa HLL, ILE usa o grupo de ativação que foi especificado quando o programa foi criado. Esse atributo é especificado usando o parâmetro de grupo de ativação (ACTGRP) no comando Criar Programa (CRTPGM) ou no comando Criar Programa de Serviço (CRTSRVPGM). Se uma ativação de programa já existir dentro do grupo de ativação indicado com esse parâmetro, ela será usada. Se o programa nunca tiver sido ativado dentro deste grupo de ativação, ele será ativado primeiro e depois executado. Se houver um grupo de ativação nomeado, o nome poderá ser alterado com o parâmetro ACTGRP nos comandos UPDPGM e UPDSRVPGM

| Uma vez que um programa é ativado, ele permanece ativado até que o grupo de ativação seja excluído. Como resultado disso,

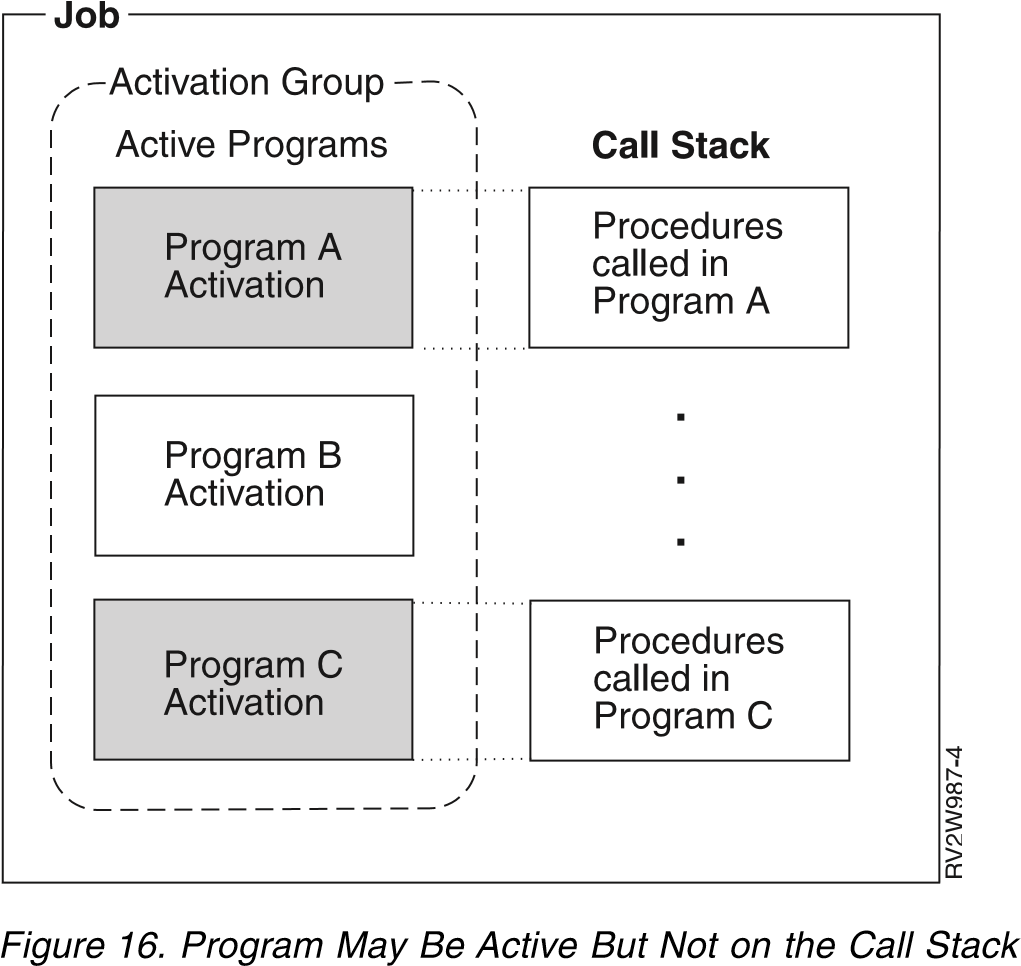
| regra, é possível ter programas ativos que não estão na pilha de chamadas. A Figura 16 mostra um exemplo de

| três programas ativos dentro de um grupo de ativação, mas apenas dois dos três programas têm procedimentos em

| a pilha de chamadas. Neste exemplo, o programa A chama o programa B, fazendo com que o programa B seja ativado. Programa B

| em seguida, retorna ao programa A. O programa A, em seguida, chama o programa C. A pilha de chamadas resultante contém procedimentos | para os programas A e C, mas não para o programa B. Para uma discussão sobre a pilha de chamadas, consulte "Pilha de chamadas" na página

| 91.



**Grupo de Ativação**

Todos os programas ILE e programas de serviço são ativados dentro de uma subestrutura de um trabalho chamado **grupo de ativação**. Essa subestrutura contém os recursos necessários para executar os programas. Esses recursos se enquadram nas seguintes categorias gerais:

Variáveis estáticas do programa

Armazenamento dinâmico

Recursos temporários de gerenciamento de dados

Certos tipos de manipuladores de exceção e procedimentos de encerramento

Os grupos de ativação usam armazenamento de nível único ou teraspace para fornecer armazenamento para variáveis estáticas de programa. Para obter mais informações, consulte o Capítulo 4, "Teraspace e armazenamento de nível único", na página 45. Quando o armazenamento de nível único é usado, as variáveis estáticas do programa e o armazenamento dinâmico recebem espaços de endereço separados para cada grupo de ativação, o que fornece algum grau de isolamento do programa e proteção contra acesso acidental. Quando o teraspace é usado, as variáveis estáticas do programa e o armazenamento dinâmico podem receber intervalos de endereços separados dentro do teraspace, o que fornece um menor grau de isolamento do programa e proteção contra acesso acidental.

Os recursos temporários de gerenciamento de dados incluem o seguinte:

Abrir arquivos (caminho de dados aberto ou ODP)

Definições de compromisso

Cursores SQL locais

Cursores SQL remotos

Sistema de arquivos hierárquico (HFS)

Gerenciador de interface do usuário

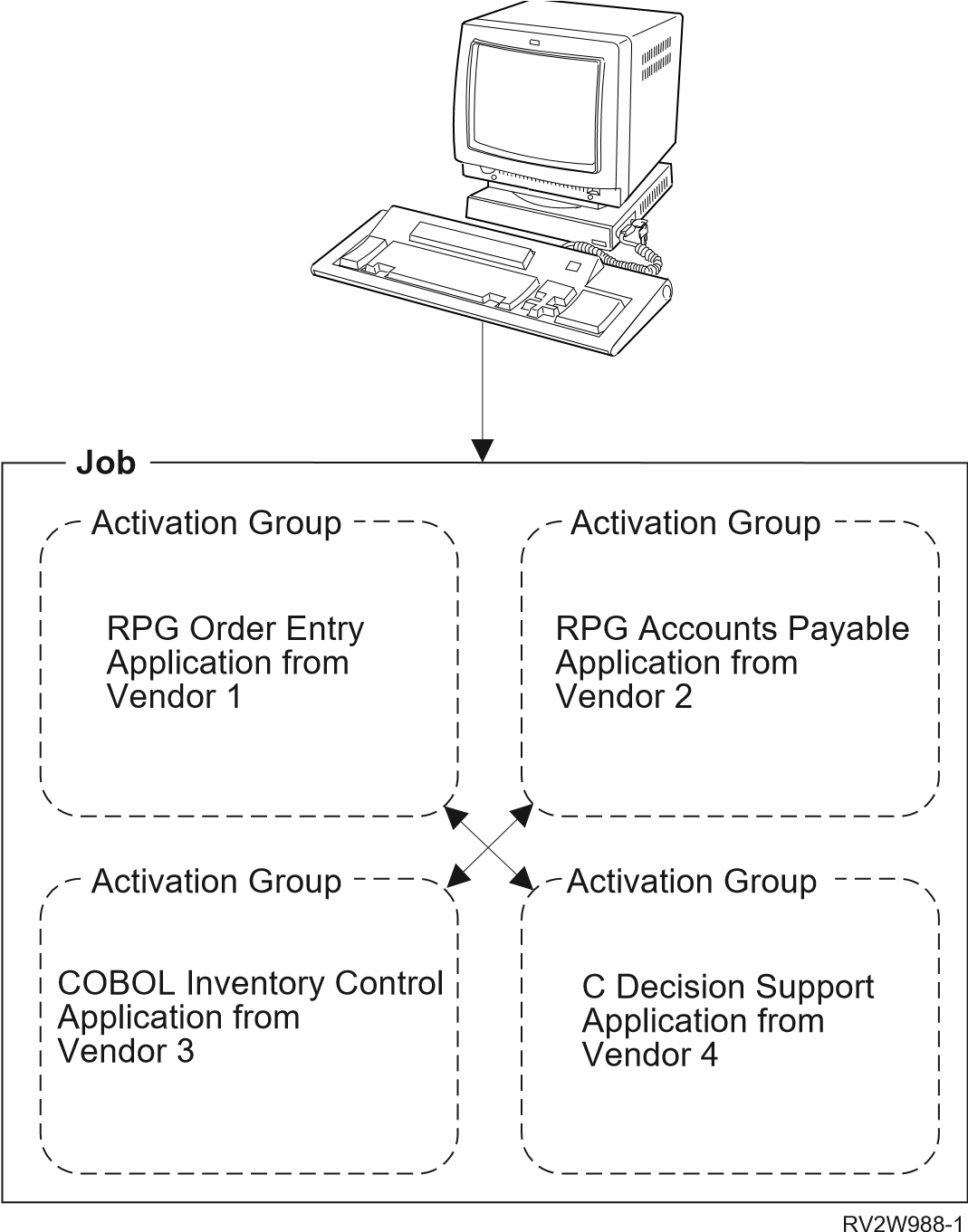
Instâncias de gerenciamento de consultas

Links de comunicação abertos

Comunicações da Common Programming Interface (CPI)

A separação desses recursos entre os grupos de ativação suporta um conceito fundamental. Ou seja, o conceito de que todos os programas ativados dentro de um grupo de ativação são desenvolvidos como um aplicativo cooperativo.

Os fornecedores de software podem selecionar diferentes grupos de ativação para isolar seus programas de outros aplicativos de fornecedores em execução no mesmo trabalho. Esse isolamento de fornecedor é mostrado na Figura 17 na página 26. Nesta figura, uma solução completa para o cliente é fornecida pela integração de pacotes de software de quatro fornecedores diferentes. Os grupos de ativação aumentam a facilidade de integração isolando os recursos associados a cada pacote de fornecedor.



*Figura 17. Grupos de ativação isolam o aplicativo de cada fornecedor*

| Há uma consequência significativa de atribuir os recursos acima a um grupo de ativação. O

| a consequência é que, quando um grupo de ativação é excluído, todos os recursos acima são retornados ao

| sistema. Os recursos temporários de gerenciamento de dados deixados abertos no momento em que o grupo de ativação é excluído | são fechados pelo sistema. O armazenamento para armazenamento estático e dinâmico que não foi desalocado é | retornou ao sistema.

**Criação de Grupo de Ativação**

| Você pode controlar a criação de tempo de execução de um grupo de ativação não padrão especificando um grupo de ativação

| quando você cria seu programa ou programa de serviço. O atributo é especificado usando o | ACTGRP no comando CRTPGM ou no comando CRTSRVPGM. Não há | Criar Ativação Comando Agrupar.

Todos os programas ILE têm um dos seguintes atributos de grupo de ativação:

v Um grupo de ativação nomeado pelo usuário

| Especificado com o parâmetro ACTGRP(name). Esse atributo permite que você gerencie uma coleção de ILE

| programas e programas de serviço como um aplicativo. O grupo de ativação é criado quando é | Necessário. Em seguida, ele é usado por todos os programas e programas de serviço que especificam o mesmo grupo de ativação

| nome.

v Um grupo de ativação nomeado pelo sistema

| Especificado com o parâmetro ACTGRP(\*NEW) no comando CRTPGM. Este atributo permite que você

| criar um novo grupo de ativação sempre que o programa for chamado. O ILE seleciona um nome para essa ativação

| grupo. O nome atribuído pelo ILE é exclusivo dentro do seu trabalho. O nome atribuído a um | nomeado pelo sistema o grupo de ativação não corresponde a nenhum nome escolhido para um grupo de ativação nomeado pelo usuário. | de serviço programas não oferecem suporte a esse atributo.

v Um atributo para usar o grupo de ativação do programa de chamada

| Especificado com o parâmetro ACTGRP(\*CALLER). Esse atributo permite que você crie um programa ILE

| ou programa de serviço que será ativado dentro do grupo de ativação do programa de chamada. Com este | , um novo grupo de ativação nunca é criado quando o programa ou programa de serviço é ativado.

v Um atributo para escolher o grupo de ativação apropriado para a linguagem de programação e armazenamento

modelo.

Especificado com o parâmetro ACTGRP(\*ENTMOD) no comando CRTPGM. Quando

ACTGRP(\*ENTMOD) é especificado, o módulo de procedimento de entrada do programa especificado pelo parâmetro ENTMOD é examinado. Ocorre uma das seguintes situações:

| – Se o atributo do módulo for RPGLE, CBLLE ou CLLE, e

| - se STGMDL(\*SNGLVL) for especificado, então QILE é usado como o grupo de ativação.

| - se STGMDL(\*TERASPACE) é especificado, então QILETS é usado como o grupo de ativação.

– Se o atributo do módulo não for RPGLE, CBLLE ou CLLE, \*NEW será usado como o grupo de ativação. ACTGRP(\*ENTMOD) é o valor padrão para esse parâmetro do comando CRTPGM.

Todos os grupos de ativação dentro de um trabalho têm um nome. Quando um grupo de ativação existe dentro de um trabalho, ele é usado para ativar programas e programas de serviço que especificam esse nome. Como resultado desse design, nomes de grupo de ativação duplicados não podem existir dentro de um trabalho.

| O parâmetro ACTGRP nos comandos UPDPGM e UPDSRVPGM pode ser usado para alterar o | grupo de ativação no qual o programa ou programa de serviço está ativado.

**Grupos de ativação padrão**

| Quando um trabalho é iniciado, o sistema cria dois grupos de ativação a serem usados por todos os programas OPM. Um dos

| esses grupos de ativação são usados para programas de aplicativo. O outro é usado para | do sistema operacional Programas. Esses grupos de ativação padrão do OPM usam armazenamento de nível único para variáveis de programa estáticas. Você | não é possível excluir os grupos de ativação padrão do OPM. Eles são excluídos pelo sistema quando seu trabalho termina.

| Os programas ILE e os programas de serviço podem ser ativados nos grupos de ativação padrão do OPM se o | São satisfeitas as seguintes condições:

| v Os programas ILE ou programas de serviço foram criados com o grupo de ativação \*CALLER opção ou com | a opção DFTACTGRP(\*YES).

| **Nota:** A opção DFTACTGRP(\*YES) só está disponível no CRTBNDCL (ILE CL) e CRTBNDRPG

| (ILE RPG) comandos.

| v A chamada para os programas ILE ou programas de serviço se origina nos grupos de ativação padrão do OPM. v O programa ILE ou o programa de serviço não usa o modelo de armazenamento teraspace.

| O sistema operacional também criará um grupo de ativação padrão teraspace quando determinar que um deles está | Necessário. O grupo de ativação padrão do teraspace usa o armazenamento teraspace para variáveis estáticas do programa. Você | não é possível excluir o grupo de ativação padrão teraspace. Ele será excluído pelo sistema quando seu trabalho terminar. | Programas ILE e programas de serviço podem ser ativados no grupo de ativação padrão teraspace se o | São satisfeitas as seguintes condições:

| v O programa ILE ou programa de serviço foi criado com o grupo de ativação \*CALLER opção.

| v O estado do programa ILE ou do programa de serviço é \*USER.

| Uma das seguintes condições também deve ser satisfeita para que o programa ILE ou o programa de serviço seja | ativado no grupo de ativação padrão do teraspace:

| v A chamada para o programa ILE ou programa de serviço origina-se no grupo de ativação padrão teraspace e

| o programa ILE ou o programa de serviço foi criado com o modelo de armazenamento \*INHERIT ou o | modelo de armazenamento \*opção TERASPACE.

| v O programa ILE ou programa de serviço foi criado com o modelo de armazenamento \*INHERIT opção, existem | nenhuma entrada de aplicativo na pilha de chamadas associada a um grupo de ativação diferente e a ativação | ocorre em preparação para uma destas invocações:

| – Procedimento armazenado SQL

| Função – SQL

| – Gatilho SQL

| **Nota:** A partir do IBM i 7.1, os procedimentos, funções e gatilhos SQL são criados com o modelo de armazenamento

| \*HERDAR. Em versões anteriores, procedimentos, funções e gatilhos SQL foram criados com

| modelo de armazenamento \*SNGLVL.

| v O programa ILE ou programa de serviço foi criado com o modelo de armazenamento \*TERASPACE opção e | não há entradas de pilha de chamadas associadas a um grupo de ativação de modelo de armazenamento teraspace. Ver | "Selecionando um grupo de ativação compatível" na página 47 para obter informações adicionais.

| O armazenamento estático e de pilha usado por programas ILE ativados em um dos grupos de ativação padrão são

| não retornou ao sistema até que o trabalho termine. Da mesma forma, recursos temporários de gerenciamento de dados associados

| com qualquer grupo de ativação padrão são normalmente definidos para um trabalho. Por exemplo, normalmente, os arquivos abertos não são | fechado pelo sistema até que o trabalho termine; consulte "Reclaim Resources Command for ILE Programs" na página 88 | para mais informações.

**Exclusão de grupo de ativação não padrão**

| Os grupos de ativação exigem que os recursos sejam criados dentro de um trabalho. O tempo de processamento pode ser economizado se um

| o grupo de ativação pode ser reutilizado por um aplicativo. O ILE oferece várias opções para permitir que você retorne

| de uma invocação sem encerrar ou excluir o grupo de ativação associado. Se a ativação | grupo é excluído depende do tipo de grupo de ativação e do método no qual o aplicativo | terminado.

| Um aplicativo pode retornar a uma entrada de pilha de chamadas (consulte "Pilha de chamadas" na página 91) associada a outro | grupo de ativação das seguintes maneiras:

v HLL verbos finais

Por exemplo, STOP RUN em COBOL ou exit() em C.

| v Chamada para API CEETREC v Exceções não tratadas

Exceções não tratadas podem ser movidas pelo sistema para uma entrada de pilha de chamadas em outro grupo de ativação. v Instruções de retorno HLL específicas do idioma

Por exemplo, uma instrução return em C, uma instrução EXIT PROGRAM em COBOL ou uma instrução RETURN em RPG. v Pular operações

| Por exemplo, enviar uma mensagem de exceção ou ramificação para uma entrada de pilha de chamadas que não esteja associada | com o seu grupo de ativação.

| Você pode excluir um grupo de ativação do seu aplicativo usando verbos de fim HLL ou chamando a API

| CEETREC. Uma exceção não tratada também pode fazer com que seu grupo de ativação seja excluído. Estas operações

| sempre excluirá seu grupo de ativação, desde que o limite de controle mais próximo seja a pilha de chamadas mais antiga

| entrada associada ao grupo de ativação (às vezes chamado de *limite de controle rígido*). Se o controle mais próximo | limite não é a entrada de pilha de chamadas mais antiga (às vezes chamada de limite de controle *suave*), o controle passa para o | entrada de pilha de chamadas antes do limite de controle. No entanto, o grupo de ativação não é excluído.

Um limite de controle é uma entrada de pilha de chamadas que representa um limite para seu aplicativo. O ILE define limites de controle sempre que você chama entre grupos de ativação. Consulte "Limites de controle" na página 32 para obter uma definição de um limite de controle.

| Um grupo de ativação nomeado pelo usuário pode ser deixado no trabalho para uso posterior. Para este tipo de grupo de ativação, qualquer

| O retorno normal ou a operação de pular além de um limite de controle rígido não exclui o grupo de ativação. Em

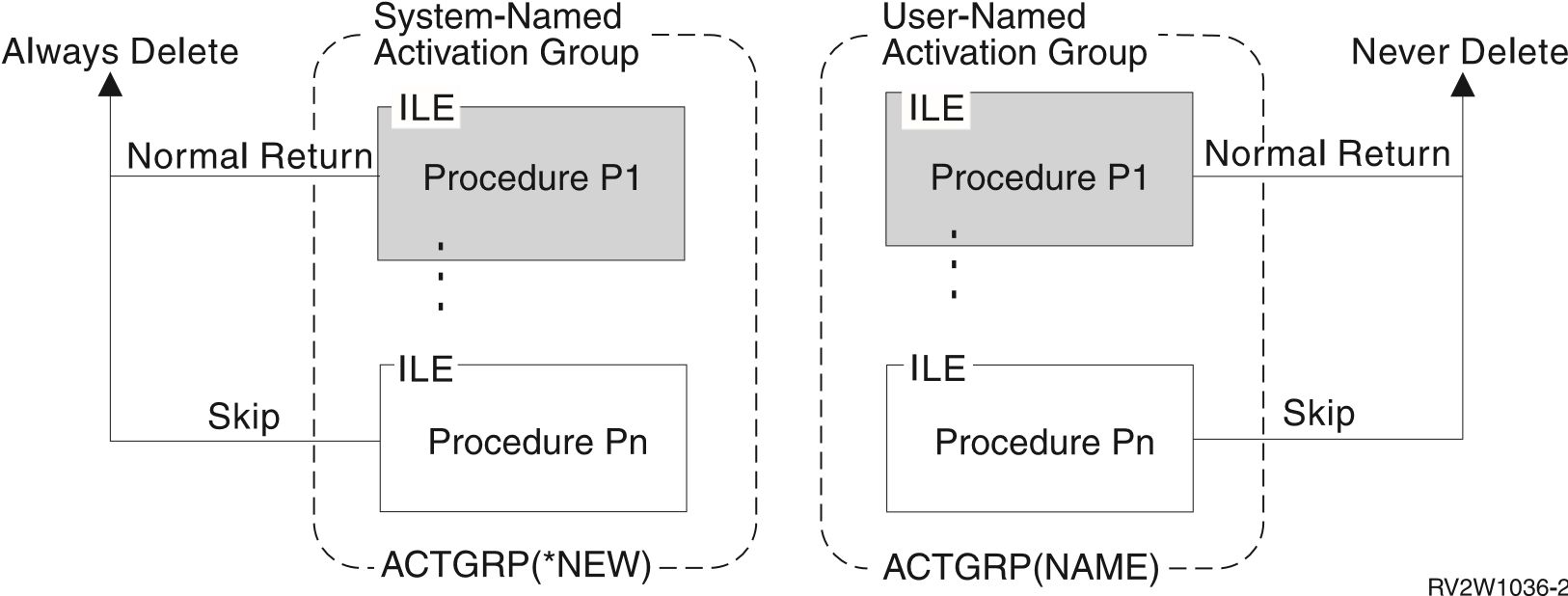
| por outro lado, o uso dessas mesmas operações dentro de um grupo de ativação nomeado pelo sistema causa a ativação

| grupo a ser excluído. Os grupos de ativação nomeados pelo sistema são excluídos porque você não pode reutilizá-los | especificando o nome gerado pelo sistema. Para regras dependentes do idioma sobre um retorno normal do | a entrada de pilha de chamadas mais antiga associada a um grupo de ativação, consulte os guias do programador ILE HLL.

| A Figura 18 mostra exemplos de como sair de um grupo de ativação. Na figura, o procedimento P1 é o mais antigo

| entrada de pilha de chamadas. Para o grupo de ativação nomeado pelo sistema (criado com a opção ACTGRP(\*NEW)), um

| o retorno normal de P1 exclui o grupo de ativação associado. Para o grupo de ativação nomeado pelo usuário | (criado com a opção ACTGRP(name)), um retorno normal de P1 não exclui o | associado grupo de ativação.



*Figura 18. Deixando grupos de ativação nomeados pelo usuário e pelo sistema*

Se um grupo de ativação nomeado pelo usuário for deixado no trabalho, você poderá excluí-lo usando o comando Recuperar Grupo de Ativação (RCLACTGRP). Esse comando permite que você exclua grupos de ativação nomeados depois que seu aplicativo retornar. Somente os grupos de ativação que não estão em uso podem ser excluídos com esse comando.

A Figura 19 na página 30 mostra um trabalho com um grupo de ativação que não está em uso e um grupo de ativação que está em uso no momento. Um grupo de ativação é considerado em uso se houver entradas de pilha de chamadas associadas a programas ativados nesse grupo de ativação. Usando o comando RCLACTGRP no programa A ou programa B exclui o grupo de ativação para o programa C e programa D.

**Trabalho**

Grupo de ativação em uso

Programas Ativos

Programas Ativos

Grupo de ativação não em uso

Procedimentos

chamado em

Programa A

Procedimentos

chamado em

Programa B

**Pilha de chamadas |**

RV2W990-4

Programa C

Ativação

Programa D

Ativação

Programa B

Programa A

*Figura 19. Grupos de ativação em uso têm entradas na pilha de chamadas*

Quando um grupo de ativação é excluído pelo ILE, ocorre determinado processamento de operação final. Esse processamento inclui a chamada de procedimentos de saída registrados pelo usuário, a limpeza do gerenciamento de dados e a limpeza de idioma (como o fechamento de arquivos). Consulte "Regras de escopo de gerenciamento de dados" na página 41 para obter detalhes sobre o processamento de gerenciamento de dados que ocorre quando um grupo de ativação é excluído.

**Ativação do programa de serviço**

O sistema segue etapas exclusivas para ativar um programa de serviço. As etapas comuns usadas para programas e programas de serviço são descritas em "Ativação do programa" na página 23. As seguintes atividades de ativação são exclusivas para programas de serviço destinados à ativação imediata:

v A ativação do programa de serviço é iniciada indiretamente como parte de uma chamada dinâmica do programa para um programa ILE.

v A ativação do programa de serviço inclui a conclusão de conexões de vinculação entre programas mapeando os links simbólicos em links físicos. v A ativação do programa de serviço inclui o processamento de verificação de assinatura.

Essas atividades de ativação são executadas para programas de serviço que estão vinculados à ativação adiada quando um de seus procedimentos importados é executado.

Um programa ativado pela primeira vez em um grupo de ativação é verificado quanto à associação a qualquer programa de serviço. Se os programas de serviço estiverem vinculados à ativação imediata do programa ativado, eles também serão ativados como parte do mesmo processamento dinâmico de chamadas. Se os programas de serviço estiverem vinculados à ativação diferida para o programa que está ativado, aqueles que satisfazem as importações de procedimento podem não ser ativados até que um de seus procedimentos importados seja chamado. Aqueles que satisfazem as importações de dados são, pelo menos, parcialmente ativados para inicializar os dados estáticos. Esse processo é repetido até que todos os programas de serviço necessários sejam ativados.

| A Figura 20 na página 31 mostra o programa ILE A vinculado aos programas de serviço B, C e D. Programas de serviço B

| e C também estão vinculados ao programa de serviço E. O atributo de grupo de ativação para cada programa e serviço

| programa é mostrado.

ILE

ILE

ILE

ILE

ILE

Programa A

ACTGRP(X)

Programa de Serviço B

ACTGRP(X)

Programa de Serviço C

ACTGRP(X)

Programa de Serviço D

ACTGRP(Y)

Programa de Serviço E

ACTGRP(\*CALLER)

RV2W991-1

*Figura*

*20*

*. Programa de Serviço*

*Ativação*

Considere o caso de que todos esses programas de serviço estão destinados à ativação imediata. Quando o programa ILE A é ativado, as seguintes ações ocorrem:

v Os programas de serviço estão localizados usando um nome de biblioteca explícito ou usando a lista de bibliotecas atual. Essa opção é controlada por você no momento em que os programas e programas de serviço são criados.

v Assim como os programas, uma ativação de programa de serviço ocorre apenas uma vez dentro de um grupo de ativação. Na Figura 20, o programa de serviço E é ativado apenas uma vez, mesmo que seja usado pelos programas de serviço B e C.

v Um segundo grupo de ativação (Y) é criado para o programa de serviço D. v A verificação de assinatura ocorre entre todos os programas e programas de serviço.

Conceitualmente, esse processo pode ser visto como a conclusão do processo de vinculação iniciado quando os programas e programas de serviço foram criados. O comando CRTPGM e o comando CRTSRVPGM salvaram o nome e a biblioteca de cada programa de serviço referenciado. Um índice em uma tabela de procedimentos exportados e itens de dados também foi salvo no programa cliente ou no programa de serviço no momento da criação do programa. O processo de ativação do programa de serviço conclui a etapa de vinculação alterando essas referências simbólicas em endereços que podem ser usados em tempo de execução.

Depois que um programa de serviço é ativado, chamadas de procedimento estático e referências de item de dados estáticos a um módulo dentro de um programa de serviço diferente são processadas. A quantidade de processamento é a mesma que seria necessária se os módulos tivessem sido vinculados por cópia ao mesmo programa. No entanto, os módulos vinculados por cópia exigem menos tempo de ativação de processamento do que os programas de serviço.

| A ativação de programas e programas de serviço requer autoridade de execução para o programa ILE e todos

| objetos de programa de serviço acoplados. Na Figura 20, a autoridade atual do chamador do programa A é usada para

| verificar a autoridade para o programa A e todos os programas de serviço vinculados. A autoridade do programa A também é | usado para verificar a autoridade de todos os programas de serviço vinculados. Observe que a autoridade do programa de serviço | B, C ou D não é usado para verificar a autoridade para o programa de serviço E.

Reconsidere a Figura 20 para o caso de o programa A se vincular aos programas de serviço B e D para ativação adiada e C para ativação imediata. D satisfaz uma importação de dados para A. B satisfaz as importações de procedimento para A apenas para chamadas de procedimento estático, mas não para quaisquer chamadas de ponteiro de procedimento. Em seguida, B se liga ao programa de serviço E para ativação diferida, enquanto C se liga a E para ativação imediata. Quando o programa A é ativado, as seguintes ações ocorrem:

v Os programas de serviço C, D e E estão localizados usando um nome de biblioteca explícito ou usando a lista de bibliotecas atual. Você pode especificar essa opção ao criar os programas e os programas de serviço. D está localizado porque satisfaz uma importação de dados e deve ser ativado pelo menos até o ponto em que seus dados estáticos são inicializados.

v E é ativado em nome de C. Quando B é executado e chama um procedimento em E, E não é ativado novamente porque uma ativação do programa de serviço ocorre apenas uma vez com um grupo de ativação.

v Um segundo grupo de ativação (Y) é criado para o programa de serviço D.

v A verificação de assinatura ocorre entre todos os programas e os programas de serviço que estão destinados à ativação imediata ou que exigem ativação parcial ou total e imediata quando A é ativado. Neste exemplo, a verificação de assinatura é feita para C, D e E.

A localização do programa de serviço B e a execução de uma verificação de assinatura para B não ocorrem até que um de seus procedimentos importados seja chamado.

A autoridade atual do chamador do programa A é usada para verificar a autoridade para o programa A e os programas de serviço C, D e E. A autoridade atual do chamador do programa de serviço B é usada para verificar a autoridade para B. A verificação de autoridade de B pode produzir resultados diferentes do que para o caso em que B está destinado à ativação imediata.

**Limites de controle**

| O ILE executa a seguinte ação quando ocorre uma verificação de função não tratada, um verbo final HLL é usado ou | API CEETREC é chamado. O ILE transfere o controle para o chamador da entrada da pilha de chamadas que representa um | para seu aplicativo. Essa entrada de pilha de chamadas é conhecida como um **limite de controle**.

Há duas definições para um limite de controle. "Limites de controle para grupos de ativação" e "Limites de controle entre entradas de pilha de chamadas OPM e ILE" na página 33 ilustram as definições a seguir.

Um limite de controle pode ser um dos seguintes:

| v Qualquer entrada de pilha de chamadas ILE para a qual a entrada de pilha de chamadas imediatamente anterior esteja associada a um | grupo de ativação diferente.

| v Qualquer entrada de pilha de chamadas ILE para a qual a entrada de pilha de chamadas imediatamente anterior é para um programa OPM.

| **Limites de controle para grupos de ativação**

Este exemplo mostra como os limites de controle são definidos entre grupos de ativação.

| A Figura 21 na página 33 mostra dois grupos de ativação e os limites de controle estabelecidos pelos vários

| Chamadas. Invocações dos procedimentos P2, P3 e P6 são potenciais limites de controle. Por exemplo, quando você | estão executando o procedimento P7, o procedimento P6 é o limite de controle. Quando você está executando procedimentos P4 ou | P5, o procedimento P3 torna-se o limite de controle.

|  |  |
| --- | --- |
| Procedimento P1 | |
|  |  |

**Chamar pilha** ILE

Grupo de Ativação A1

RV2W992-3

Procedimento P7

Procedimento P5

Processo P4

Grupo de Ativação A2

ILE

ILE

ILE

ILE

ILE

ILE

Procedimento P6

Procedimento P3

Procedimento P2

*Figura 21. Limites de controle.* Os procedimentos sombreados são limites de controle.

| **Nota:** Os limites de controle são definidos entre todos os grupos de ativação, incluindo a ativação padrão

| Grupos. Por exemplo, se uma entrada de pilha de chamadas associada ao grupo de ativação padrão teraspace | precede imediatamente uma entrada de pilha de chamadas associada a um grupo de ativação padrão do OPM, | ser um limite de controle entre as duas entradas da pilha de chamadas.

**Limites de controle entre as entradas de pilha de chamadas OPM e ILE**

| Este exemplo mostra como os limites de controle são definidos entre as entradas da pilha de chamadas ILE e as | de chamadas do OPM entradas de pilha.

| A Figura 22 na página 34 mostra três procedimentos ILE (P1, P2 e P3) em execução em uma ativação padrão do OPM

| grupo. Este exemplo poderia ter sido criado usando o comando CRTPGM ou CRTSRVPGM

| com o valor do parâmetro ACTGRP(\*CALLER). Invocações dos procedimentos P1 e P3 são | limites de controle potenciais porque as entradas de pilha de chamadas anteriores estão associadas a programas OPM | A e B.

Grupo de Ativação Padrão

\*DFTACTGRP

RV2W1040-1

Programa A

Programa B

Procedimento P2

Procedimento P3

Procedimento P1

ILE

OPM

OPM

ILE

ILE

*Figura 22. Controle os limites entre as entradas da pilha de chamadas ILE e as entradas da pilha de chamadas OPM.* Os procedimentos sombreados são limites de controle.

**Uso do limite de controle**

| Quando você usa um verbo final ILE HLL ou chama API CEETREC, ILE usa o limite de controle mais recente em | a pilha de chamadas para determinar para onde transferir o controle. A entrada da pilha de chamadas imediatamente antes do controle | o limite recebe o controle depois que o ILE conclui todo o processamento final.

O limite de controle é usado quando uma verificação de função não tratada ocorre dentro de um procedimento ILE. O limite de controle define o ponto na pilha de chamadas no qual a verificação de função não tratada é **promovida** à condição de falha ILE genérica. Para obter informações adicionais, consulte "Tratamento de erros".

| Quando o limite de controle mais próximo é a entrada de pilha de chamadas mais antiga associada a uma ativação não padrão

| group, qualquer verbo final HLL, chamada para a API CEETREC ou verificação de função não tratada causa a ativação

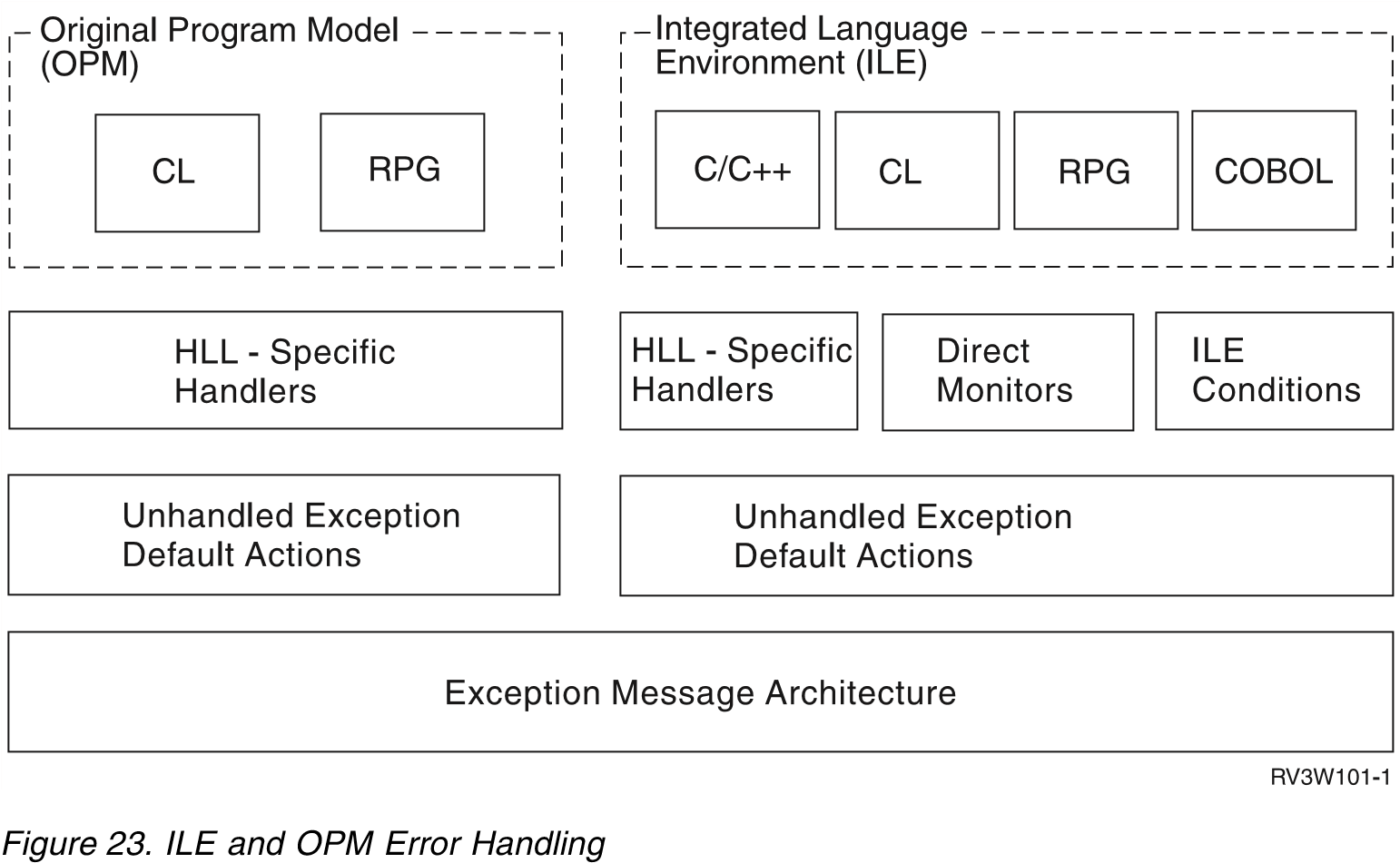
| grupo a ser excluído. Quando o limite de controle mais próximo não é a entrada de pilha de chamadas mais antiga associada ao | um grupo de ativação não padrão, o controle retorna à entrada da pilha de chamadas imediatamente antes do limite de controle.

| O grupo de ativação não é excluído porque existem entradas de pilha de chamadas anteriores para o mesmo grupo de ativação.

| A Figura 21 na página 33 mostra o procedimento P2 e o procedimento P3 como as entradas de pilha de chamadas mais antigas associadas | com seus grupos de ativação. Usando um verbo final HLL ou chamando a API CEETREC no procedimento P2, P3, P4 ou | P5 (mas não P6 ou P7) faria com que o grupo de ativação A2 fosse excluído.

**Tratamento de erros**

Este tópico explica os recursos avançados de tratamento de erros para programas OPM e ILE. Para entender como esses recursos se encaixam na arquitetura de mensagem de exceção, consulte a Figura 23 na página 35. Informações de referência específicas e conceitos adicionais são encontrados no Capítulo 9, "Gerenciamento de Exceções e Condições", na página 105. A Figura 23 na página 35 mostra uma visão geral do tratamento de erros. Este tópico começa com a camada inferior desta figura e continua até a camada superior. A camada superior representa as funções que você pode usar para lidar com erros em um programa OPM ou ILE.



**Filas de mensagens de trabalho**

Existe uma fila de mensagens para cada entrada de pilha de chamadas dentro de cada trabalho. Essa fila de mensagens facilita o envio e o recebimento de mensagens informativas e mensagens de exceção entre os programas e procedimentos em execução na pilha de chamadas. A fila de mensagens é chamada de **fila de mensagens de chamada**.

A fila de mensagens de chamada é identificada pelo nome do programa OPM ou do procedimento ILE que está na pilha de chamadas. O nome do procedimento ou o nome do programa pode ser usado para especificar a entrada da pilha de chamadas de destino para a mensagem que você envia. Como os nomes de procedimento ILE não são exclusivos, o nome do módulo ILE e o nome do programa ou programa de serviço ILE podem opcionalmente ser especificados. Quando o mesmo programa ou procedimento tem várias entradas de pilha de chamadas, a fila de mensagens de chamada mais próxima é usada.

Além das filas de mensagens de chamada, cada trabalho contém uma fila de mensagens externas. Todos os programas e procedimentos em execução no trabalho podem enviar e receber mensagens entre um trabalho interativo e o usuário da estação de trabalho usando essa fila.

Para obter informações sobre como enviar e receber mensagens de exceção usando APIs, consulte a Mensagem

Manipulando APIs na coleção de tópicos API da categoria Programação no Centro de Informações do IBM i.

**Mensagens de exceção e como elas são enviadas**

Este tópico descreve os diferentes tipos de mensagem de exceção e as maneiras pelas quais uma mensagem de exceção pode ser enviada.

O tratamento de erros para ILE e OPM é baseado em tipos de mensagem de exceção. A menos que qualificado de outra forma, o termo mensagem de **exceção** indica qualquer um destes tipos de mensagem:

#### Escape (\*ESCAPE)

Indica um erro que faz com que um programa termine de forma anormal, sem concluir seu trabalho. Você não receberá controle depois de enviar uma mensagem de exceção de escape.

#### Status (\*STATUS)

Descreve o status do trabalho que está sendo feito por um programa. Você receberá controle depois de enviar esse tipo de mensagem. Se você recebe controle depende da maneira como o programa de recebimento manipula a mensagem de status.

#### Notificar (\*NOTIFICAR)

Descreve uma condição que requer ação corretiva ou uma resposta do programa de chamada. Você receberá controle depois de enviar esse tipo de mensagem. Se você recebe controle depende da maneira como o programa de recebimento manipula a mensagem de notificação.

#### Verificação de função

Descreve uma condição final que não foi esperada pelo programa. Uma verificação de função ILE, CEE9901, é um tipo de mensagem especial que é enviada apenas pelo sistema. Uma verificação de função OPM é um tipo de mensagem de escape com um ID de mensagem CPF9999.

Para obter informações sobre estes tipos de mensagem e outros tipos de mensagem, consulte a coleção de tópicos API da categoria Programação do Centro de Informações do IBM i.

Uma mensagem de exceção é enviada das seguintes maneiras:

v Gerado pelo sistema

O sistema operacional (incluindo seu HLL) gera uma mensagem de exceção para indicar um erro de programação ou informações de status.

v API do manipulador de mensagens

A API de Enviar Mensagem de Programa (QMHSNDPM) pode ser usada para enviar uma mensagem de exceção para uma fila de mensagens de chamada específica.

v API ILE

A API vinculável Signal a Condition (CEESGL) pode ser usada para gerar uma condição ILE. Essa condição resulta em uma mensagem de exceção de escape ou mensagem de exceção de status.

v Verbos específicos do idioma

Para ILE C e ILE C++, a função raise() gera um sinal C. Nem o ILE RPG nem o ILE COBOL têm uma função semelhante.

**Como as mensagens de exceção são manipuladas**

Quando você ou o sistema enviam uma mensagem de exceção, o processamento de exceção é iniciado. Esse processamento continua até que a exceção seja manipulada, que é quando a mensagem de exceção é modificada para indicar que ela foi manipulada.

O sistema modifica a mensagem de exceção para indicar que ela foi manipulada quando chama um manipulador de exceção para uma fila de mensagens de chamada OPM. O ILE HLL modifica a mensagem de exceção antes que o manipulador de exceções seja chamado para uma fila de mensagens de chamada ILE. Como resultado, o tratamento de erros específico do HLL considera a mensagem de exceção manipulada quando o manipulador é chamado. Se você não usar o tratamento de erros específico do HLL, o HLL do ILE poderá manipular a mensagem de exceção ou permitir que o processamento de exceção continue. Consulte o manual de referência do ILE HLL para determinar as ações padrão do HLL para mensagens de exceção não tratadas.

Com recursos adicionais definidos para o ILE, você pode ignorar o tratamento de erros específicos do idioma. Esses recursos incluem manipuladores de monitor diretos e manipuladores de condição ILE. Ao usar esses recursos, você é responsável por alterar a mensagem de exceção para indicar que a exceção é tratada. Se você não alterar a mensagem de exceção, o sistema continuará o processamento de exceção tentando localizar outro manipulador de exceções. O tópico "Tipos de manipuladores de exceção" na página 38 contém detalhes sobre manipuladores de monitor direto e manipuladores de condição ILE. Para obter informações que explicam como alterar uma mensagem de exceção, consulte a API Change Exception Message (QMHCHGEM) na coleção de tópicos API da categoria Programação do Centro de Informações do IBM i.

**Recuperação de exceção**

Talvez você queira continuar o processamento depois que uma exceção for enviada. A recuperação de um erro pode ser uma ferramenta de aplicativo útil que permite que você entregue aplicativos que toleram erros. Para programas ILE e OPM, o sistema definiu o conceito de um **ponto de currículo**. O ponto de retomada é inicialmente definido como uma instrução imediatamente após a ocorrência da exceção. Depois de lidar com uma exceção, você pode continuar o processamento em um ponto de retomada. Para obter mais informações sobre como usar e modificar um ponto de currículo, consulte o Capítulo 9, "Gerenciamento de exceções e condições", na página 105.

**Ações padrão para exceções não tratadas**

Se você não manipular uma mensagem de exceção em seu HLL, o sistema executará uma ação padrão para a exceção não tratada.

A Figura 23 na página 35 mostra as ações padrão para exceções não tratadas com base no fato de a exceção ter sido enviada a um programa OPM ou ILE. Diferentes ações padrão para OPM e ILE criam uma diferença fundamental nos recursos de tratamento de erros.

Para o OPM, uma exceção não tratada gera uma mensagem de escape especial conhecida como mensagem de verificação de função. Esta mensagem recebe o ID de mensagem especial do CPF9999. Ele é enviado para a fila de mensagens de chamada da entrada da pilha de chamadas que incorreu na mensagem de exceção original. Se a mensagem de verificação de função não for tratada, o sistema removerá essa entrada de pilha de chamadas. O sistema então envia a mensagem de verificação de função para a entrada de pilha de chamadas anterior. Esse processo continua até que a mensagem de verificação de função seja manipulada. Se a mensagem de verificação de função nunca for tratada, o trabalho será encerrado.

Para o ILE, uma mensagem de exceção não tratada é percolada para a fila de mensagens de entrada da pilha de chamadas anterior. **A percolação** ocorre quando a mensagem de exceção é movida para a fila de mensagens de chamada anterior. Isso cria o efeito de enviar a mesma mensagem de exceção para a fila de mensagens de chamada anterior. Quando isso acontece, o processamento de exceções continua na entrada da pilha de chamadas anterior.

| A percolação de uma exceção não tratada continua até que um limite de controle seja atingido ou o | mensagem de exceção é tratada. As etapas de processamento executadas quando uma exceção não tratada percola para um | o limite de controle depende do tipo de exceção:

|  |  |  |
| --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  | | 1.  2.  3.  4. | Se for uma exceção de status, a exceção será manipulada e o remetente do status poderá continuar.  Se for uma exceção de notificação, a resposta padrão será enviada, a exceção será manipulada e o remetente da notificação poderá continuar.  Se for uma exceção de escape, uma mensagem de verificação de função especial será enviada para o ponto de retomada. Essa verificação de função pode ser manipulada ou pode percolar para um limite de controle (veja abaixo).  Se for uma verificação de função, o ILE considerará que o aplicativo terminou com um erro inesperado. Todas as entradas de pilha de chamadas até e incluindo o limite de controle são canceladas. Se o limite de controle for a invocação mais antiga associada a um grupo de ativação não padrão, o grupo de ativação será encerrado. Uma mensagem de exceção de falha genérica é definida pelo ILE para todos os idiomas. O identificador de mensagem para esta mensagem é CEE9901 e é enviado pelo ILE para a entrada de pilha de chamadas imediatamente anterior ao limite de controle. |

| A Figura 24 na página 38 mostra uma exceção de escape não tratada dentro do ILE. Neste exemplo, a invocação de

| O procedimento P1 é um limite de controle e também é a entrada de invocação mais antiga para a ativação associada

| grupo. O procedimento P4 incorre em uma exceção de escape. A exceção de escape percola para o controle P1

| e, como resultado, a mensagem de verificação de função especial é enviada para o ponto de retomada do procedimento P3. O

| a verificação da função não é manipulada e percola para o limite de controle P1. O pedido é considerado | terminaram e o grupo de ativação é destruído. Finalmente, a exceção de escape CEE9901 é enviada para o | entrada de pilha de chamadas imediatamente anterior ao limite de controle (programa A).

A ação padrão para mensagens de exceção não tratadas definidas no ILE permite que você se recupere de condições de erro que ocorrem em um aplicativo de idioma misto. Para erros inesperados, o ILE impõe uma mensagem de falha consistente para todos os idiomas. Isso melhora a capacidade de integrar aplicativos de diferentes fontes.

Grupo de Ativação

**Pilha de chamadas |**

Processo P4

Programa A

OPM

ILE

ILE

ILE

ILE

Procedimento P2

Permear

Unhandled

Exceção

Permear

Função

Verificar

Procedimento P3

Procedimento P1

*Ponto de Currículo*

Enviar

Terminação

Exceção

CEE9901

RV2W1043-1

*Figura 24. Ação padrão de exceção não tratada*

**Tipos de manipuladores de exceção**

Este tópico fornece uma visão geral dos tipos de manipulador de exceção fornecidos para programas OPM e ILE. Como mostrado na Figura 23 na página 35, essa é a camada superior da arquitetura da mensagem de exceção. O ILE fornece recursos adicionais de tratamento de exceções quando comparado ao OPM.

Para programas OPM, o tratamento de erros específico do HLL fornece uma ou mais rotinas de manipulação para cada entrada de pilha de chamadas. A rotina apropriada é chamada pelo sistema quando uma exceção é enviada para um programa OPM.

O tratamento de erros específicos de HLL no ILE fornece os mesmos recursos. O ILE, no entanto, tem tipos adicionais de manipuladores de exceções. Esses tipos de manipuladores oferecem controle direto da arquitetura de mensagens de exceção e permitem que você ignore o tratamento de erros específicos do HLL. Os tipos adicionais de manipuladores para ILE são: Manipulador de monitor direto manipulador de condição ILE

Para determinar se esses tipos de manipuladores são suportados pelo seu HLL, consulte o guia do programador do ILE HLL.

**Os manipuladores de monitor diretos** permitem que você declare diretamente um monitor de exceção em torno de instruções de origem HLL limitadas. Para a ILE C, esse recurso é habilitado por meio de uma diretiva #pragma. O ILE COBOL não declara diretamente um monitor de exceção em torno de instruções de origem HLL limitadas no mesmo sentido que o ILE C. Um programa ILE COBOL não pode codificar diretamente a habilitação e desabilitação de manipuladores em torno de código-fonte arbitrário. No entanto, uma declaração como

ADICIONAR a a b NO ERRO DE TAMANHO imperativo

é mapeado internamente para usar o mecanismo de monitor direto, apesar de ser um manipulador específico de HLL. Assim, em termos da prioridade de qual manipulador obtém o controle primeiro, esse imperativo condicional com escopo de instrução obtém controle antes do manipulador de condição ILE (registrado por meio do CEEHDLR). O controle, em seguida, prossegue para os declarativos USE no COBOL.

**Os manipuladores de condição ILE** permitem que você **registre** um manipulador de exceção em tempo de execução. Os manipuladores de condição ILE são registrados para uma entrada de pilha de chamadas específica. Para registrar um manipulador de condição ILE, use a API vinculável Registrar um manipulador de condição escrito pelo usuário (CEEHDLR). Essa API permite que você identifique um procedimento em tempo de execução que deve receber controle quando ocorrer uma exceção. A API CEEHDLR requer a capacidade de declarar e definir um ponteiro de procedimento dentro do seu idioma. CEEHDLR é implementado como uma função interna. Portanto, seu endereço não pode ser especificado e não pode ser chamado por meio de um ponteiro de procedimento. Os manipuladores de condição ILE podem não ser **registrados** chamando a API vinculável Cancelar Registro de um Manipulador de Condição Escrito pelo Usuário (CEEHDLU).

OPM e ILE suportam manipuladores específicos de HLL. **Manipuladores específicos de HLL** são os recursos de linguagem definidos para lidar com erros. Por exemplo, a função de sinal ILE C pode ser usada para lidar com mensagens de exceção. O tratamento de erros específicos de HLL no RPG inclui a capacidade de manipular exceções para uma única instrução (extensor E), um grupo de instruções (MONITOR) ou um procedimento inteiro (\*sub-rotinas PSSR e INFSR). O tratamento de erros específicos de HLL no COBOL inclui declarativo USE para tratamento de erros de E/S e imperativos em frases de condição com escopo de instrução, como ON SIZE ERROR e AT INVALID KEY.

A prioridade do manipulador de exceções se torna importante se você usar o tratamento de erros específicos do HLL e tipos adicionais de manipulador de exceções ILE.

A Figura 25 na página 40 mostra uma entrada de pilha de chamadas para o procedimento P2. Neste exemplo, todos os três tipos de manipuladores foram definidos para uma única entrada de pilha de chamadas. Embora este possa não ser um exemplo típico, é possível ter todos os três tipos definidos. Como todos os três tipos são definidos, uma prioridade do manipulador de exceções é definida. A figura mostra essa prioridade. Quando uma mensagem de exceção é enviada, os manipuladores de exceção são chamados na seguinte ordem:

1. Manipuladores diretos de monitor

Primeiro a invocação é escolhida, depois a ordem relativa dos manipuladores nessa invocação. Dentro de uma invocação, todos os manipuladores de monitor diretos; RPG (E), MONITOR, INFSR e indicadores de erro; e os imperativos condicionais com escopo de instrução COBOL obtêm controle antes dos manipuladores de condição ILE. Da mesma forma, os manipuladores de condição ILE obtêm controle antes de outros manipuladores específicos de HLL.

Se os manipuladores de monitor direto tiverem sido declarados em torno das instruções que incorreram na exceção, esses manipuladores serão chamados antes dos manipuladores específicos do HLL. Por exemplo, se o procedimento P2 na Figura 25 na página 40 tiver um manipulador específico de HLL e o procedimento P1 tiver um manipulador de monitor direto, o manipulador de P2 será considerado antes do manipulador de monitor direto de P1.

Os monitores diretos podem ser aninhados lexicalmente. O manipulador especificado no monitor direto aninhado mais profundamente é escolhido primeiro dentro dos monitores aninhados de multiplicação que especificam o mesmo número de prioridade.

1. Manipulador de condição ILE

Se um manipulador de condição ILE tiver sido registrado para a entrada de pilha de chamadas, esse manipulador será chamado de segundo.

Vários manipuladores de condição ILE podem ser registrados. No exemplo, o procedimento P5 e o procedimento P6 são manipuladores de condição ILE. Quando vários manipuladores de condição ILE são registrados para a mesma entrada de pilha de chamadas, o sistema chama esses manipuladores na ordem ELFO (last-in-first-out). Em geral, os manipuladores específicos de HLL têm a prioridade mais baixa, depois dos manipuladores de monitor direto e dos manipuladores de condição. Exceções são os manipuladores específicos de HLL mencionados na discussão de manipuladores de monitor direto.

1. Manipulador específico de HLL

Os manipuladores específicos de HLL são chamados de últimos.

O sistema encerra o processamento de exceção quando uma mensagem de exceção é modificada para mostrar que ela foi manipulada. Se você estiver usando manipuladores de monitor direto ou manipuladores de condição ILE, modificar a mensagem de exceção é de sua responsabilidade. Várias ações de controle estão disponíveis. Por exemplo, você pode especificar handle como uma ação de controle. Enquanto a mensagem de exceção permanecer sem tratamento, o sistema continuará a procurar um manipulador de exceção usando as prioridades definidas anteriormente. Se a exceção não for tratada dentro da entrada da pilha de chamadas atual, ocorrerá a percolação para a entrada de pilha de chamadas anterior. Se você não usar o tratamento de erros específico do HLL, o HLL do ILE poderá optar por permitir que o tratamento de exceções continue na entrada da pilha de chamadas anterior.

**Pilha de chamadas |**

RV2W1041-3

Procedimento P1

ILE

ILE

ILE

ILE

ILE

ILE

ILE

Procedimento P3

Idioma padrão

Inadimplência

Monitor Direto

Treinador

Processo P4

Condição ILE

Treinador

Procedimento P5

Última em

Primeiro a sair

Condição ILE

Treinador

Procedimento P6

HLL - Específico

Treinador

Procedimento P7

Procedimento P2

Ocorre uma exceção

**Manipulador de exceções**

**Prioridade**

.

.

.

*Figura*

*25*

*. Prioridade do manipulador de exceções*

**Condições ILE**

| Para permitir maior consistência entre sistemas, o ILE definiu um recurso que você pode usar para trabalhar com erros | Condições. Uma condição ILE é uma representação independente do sistema de uma condição de erro dentro de um HLL.

| Cada condição ILE tem uma mensagem de exceção correspondente. Uma condição ILE é representada por um

| token de condição. Um *token de condição*  é uma estrutura de dados de 12 bytes que é consistente em vários | sistemas participantes. Essa estrutura de dados contém informações por meio das quais você pode associar o | com a mensagem de exceção subjacente.

Para escrever programas que são consistentes entre sistemas, você precisa usar manipuladores de condição ILE e tokens de condição ILE. Para obter mais informações sobre as condições de ILE, consulte o Capítulo 9, "Gerenciamento de exceções e condições", na página 105.

**Regras de escopo de gerenciamento de dados**

As regras de escopo de gerenciamento de dados controlam o uso de recursos de gerenciamento de dados. Esses recursos são objetos temporários que permitem que um programa trabalhe com o gerenciamento de dados. Por exemplo, quando um programa abre um arquivo, um objeto chamado ODP (caminho de dados abertos) é criado para conectar o programa ao arquivo. Quando um programa cria uma substituição para alterar como um arquivo deve ser processado, o sistema cria um objeto de substituição.

As regras de escopo de gerenciamento de dados determinam quando um recurso pode ser compartilhado por vários programas ou procedimentos em execução na pilha de chamadas. Por exemplo, arquivos abertos criados com o valor do parâmetro SHARE(\*YES) ou objetos de definição de compromisso podem ser usados por muitos programas ao mesmo tempo. A capacidade de compartilhar um recurso de gerenciamento de dados depende do nível de escopo do recurso de gerenciamento de dados.

As regras de escopo de gerenciamento de dados também determinam a existência do recurso. O sistema exclui automaticamente recursos não utilizados dentro do trabalho, dependendo das regras de escopo. Como resultado dessa operação de limpeza automática, o trabalho usa menos armazenamento e o desempenho do trabalho melhora.

O ILE formaliza as regras de escopo de gerenciamento de dados para programas OPM e ILE nos seguintes níveis de escopo:

Chamar

Trabalho do grupo de ativação

Dependendo do recurso de gerenciamento de dados que você está usando, um ou mais dos níveis de escopo podem ser explicitamente especificados. Se você não selecionar um nível de escopo, o sistema selecionará um dos níveis como padrão.

Consulte o Capítulo 11, "Escopo de gerenciamento de dados", na página 117 para obter informações sobre como cada recurso de gerenciamento de dados oferece suporte aos níveis de escopo.

**Escopo em nível de chamada**

O escopo em nível de chamada ocorre quando o recurso de gerenciamento de dados está conectado à entrada da pilha de chamadas que criou o recurso. A Figura 26 na página 42 mostra um exemplo. O escopo em nível de chamada geralmente é o nível de escopo padrão para programas executados no grupo de ativação padrão. Nesta figura, o programa OPM A, o programa B do OPM ou o procedimento P2 do ILE podem optar por retornar sem fechar seus respectivos arquivos F1, F2 ou F3. O gerenciamento de dados associa o ODP de cada arquivo ao número de nível de chamada que abriu o arquivo. O comando RCLRSC pode ser usado para fechar os arquivos com base em um número de nível de chamada específico passado para esse comando.

*Figura 26. Escopo em nível de chamada.* ODPs e substituições podem ter o escopo definido para o nível de chamada.

As substituições com escopo para um nível de chamada específico são excluídas quando a entrada da pilha de chamadas correspondente retorna. As substituições podem ser compartilhadas por qualquer entrada de pilha de chamadas que esteja abaixo do nível de chamada que criou a substituição.

**Escopo de nível de grupo de ativação**

| O escopo no nível do grupo de ativação ocorre quando o recurso de gerenciamento de dados está conectado à ativação

| grupo associado ao programa ILE ou à invocação do procedimento que criou o recurso. Quando o | o grupo de ativação é excluído, o gerenciamento de dados fecha todos os recursos associados ao grupo de ativação | que foram deixados em aberto. A Figura 27 mostra um exemplo de escopo no nível do grupo de ativação.

| O escopo no nível do grupo de ativação é o nível de escopo padrão para a maioria dos tipos de recursos de gerenciamento de dados | usado por procedimentos ILE não executados em um grupo de ativação padrão. Por exemplo, a figura mostra ODPs | para os arquivos F1, F2 e F3 e substitua R1 com escopo para o grupo de ativação.

RV3W102-0

PEP P1

UEP P2

Procedimento P3

ILE

ILE

ILE

ODP F1

ODP F2

ODP F3

Recursos de gerenciamento de dados

.

.

.

Substituir R1

Grupo de ativação ILE

*Figura 27. Escopo de Nível do Grupo de Ativação.* ODPs e substituições podem ter o escopo de um grupo de ativação.

A capacidade de compartilhar um recurso de gerenciamento de dados com escopo para um grupo de ativação é limitada a programas em execução nesse grupo de ativação. Isso fornece isolamento e proteção de aplicativos. Por exemplo, suponha que o arquivo F1 na figura foi aberto com o valor do parâmetro SHARE(\*YES). O arquivo F1 pode ser usado por qualquer procedimento ILE em execução no mesmo grupo de ativação. Outra operação aberta para o arquivo F1 em um grupo de ativação diferente resulta na criação de um segundo ODP para esse arquivo.

**Escopo de nível de trabalho**

O escopo no nível do trabalho ocorre quando o recurso de gerenciamento de dados está conectado ao trabalho. O escopo em nível de trabalho está disponível para os programas OPM e ILE. O escopo em nível de trabalho permite o compartilhamento de recursos de gerenciamento de dados entre programas em execução em diferentes grupos de ativação. Conforme descrito no tópico anterior, o escopo de recursos para um grupo de ativação limita o compartilhamento desse recurso para programas em execução nesse grupo de ativação. O escopo no nível do trabalho permite o compartilhamento de recursos de gerenciamento de dados entre todos os programas ILE e OPM em execução no trabalho.

A Figura 28 mostra um exemplo de escopo no nível do trabalho. O programa A pode ter aberto o arquivo F1, especificando o escopo em nível de trabalho. O ODP para esse arquivo está conectado ao trabalho. O arquivo não é fechado pelo sistema, a menos que o trabalho termine. Se o ODP tiver sido criado com o valor do parâmetro SHARE(YES), qualquer programa OPM ou procedimento ILE poderá potencialmente compartilhar o arquivo.

Grupo de ativação ILE

Grupo de Ativação Padrão

RV2W1039-2

Programa A

Programa B

PEP P1

UEP P2

OPM

OPM

ILE

ILE

ODP F1

Substituir R1

Compromisso

Definição \*JOB

Gerenciamento de dados

Recursos

.

.

.

**Trabalho**

*Figura 28. Escopo do nível do trabalho.* ODPs, substituições e definições de compromisso podem ter o escopo definido para o nível de trabalho.

As substituições com escopo para o nível de trabalho influenciam todas as operações de arquivo aberto no trabalho. Neste exemplo, a substituição R1 poderia ter sido criada pelo procedimento P2. Uma substituição de nível de trabalho permanece ativa até que seja explicitamente excluída ou o trabalho seja encerrado. A substituição de nível de trabalho é a substituição de prioridade mais alta quando ocorre a mesclagem. Isso ocorre porque as substituições de nível de chamada são mescladas quando existem várias substituições na pilha de chamadas.

Os níveis de escopo de gerenciamento de dados podem ser explicitamente especificados pelo uso de parâmetros de escopo em comandos de substituição, comandos de controle de compromisso e por meio de várias APIs. A lista completa de recursos de gerenciamento de dados que usam as regras de escopo está no Capítulo 11, "Escopo de Gerenciamento de Dados", na página 117.

**44** Conceitos de ILE IBM i 7.1

**Capítulo 4. Teraspace e armazenamento de nível único**

Ao criar programas ILE, você pode selecionar um dos seguintes modelos de armazenamento em alguns compiladores: v Armazenamento de nível único v Teraspace v Herdar

O modelo de armazenamento herdado indica que os programas adotarão o modelo de armazenamento, teraspace ou armazenamento de nível único, do grupo de ativação no qual eles são ativados. Os programas ILE usam armazenamento de nível único por padrão.

Este capítulo se concentra nas opções de teraspace.

**Características do Teraspace**

Teraspace é um grande espaço temporário que é local para um trabalho. Um teraespaço fornece um espaço de endereço contíguo, mas pode consistir em muitas áreas alocadas individualmente, com áreas não alocadas no meio. O teraespaço não existe mais do que o tempo entre o início e o fim do trabalho.

Um teraespaço não é um objeto espacial. Isso significa que ele não é um objeto do sistema e que você não pode se referir a ele usando um ponteiro do sistema. No entanto, o teraspace é endereçável com ponteiros de espaço dentro do mesmo trabalho.

A tabela a seguir mostra como o teraspace se compara ao armazenamento de nível único.

*Tabela 2. Comparando teraspace com armazenamento de nível único*

|  |  |  |
| --- | --- | --- |
| **Atributos** | **Teraespaço** | **Armazenamento de nível único** |
| Localidade | Processo local: normalmente acessível apenas ao trabalho de propriedade. | Global: acessível a qualquer trabalho que tenha um ponteiro para ele. |
| Tamanho | Aproximadamente 100 TB no total | Muitas unidades de 16 MB. |
| Suporta mapeamento de memória? | Sim | Não |
| Endereçado por ponteiros de 8 bytes? | Sim | Não |
| Apoia a partilha entre empregos? | Deve ser feito usando APIs de memória compartilhada (por exemplo, shmat ou mmap). | Pode ser feito passando ponteiros para outros trabalhos ou usando APIs de memória compartilhada. |

**Usando o Teraspace para armazenamento**

| O *modelo*  de armazenamento de um programa determina o tipo de armazenamento usado para armazenamento automático, estático e constante.

| Por padrão, os compiladores também usam interfaces de armazenamento de pilha que correspondem ao modelo de armazenamento. No entanto, alguns | Os compiladores permitem que o tipo de armazenamento de pilha seja escolhido independentemente do modelo de armazenamento de um programa.

| Os compiladores ILE C e C++ fornecem a opção de comando TERASPACE (\*YES \*TSIFC) create para permitir | o uso de versões teraspace de interfaces de armazenamento de pilha de programas de armazenamento de nível único sem | de origem alterações de código. Por exemplo, malloc() é mapeado para \_C\_TS\_malloc().

| O compilador ILE RPG permite que o tipo de armazenamento de pilha seja explicitamente definido, independentemente do armazenamento | , usando a palavra-chave ALLOC na especificação de controle.

© Direitos autorais IBM Corp. 1997, 2010

|

Veja o

ILE C/C++ P

[r](http://publib.boulder.ibm.com/infocenter/iseries/v6r1m0/topic/rzaha/sc092712.pdf)

Guia do ogrammer



ou

ILE RPG Pr

Guia do ogrammer



para obter detalhes sobre estes

opções do compilador.

|

**Escolhendo um modelo de armazenamento de programa**

| Opcionalmente, você pode criar seus módulos e programas ILE para que eles usem o *modelo de armazenamento teraspace*.

| Os programas de modelo de armazenamento Teraspace usam teraspace para armazenamento automático, estático e constante. Quando você | escolha o modelo de armazenamento teraspace, você pode usar áreas maiores para alguns desses tipos de armazenamento. Ver | "Usando o modelo de armazenamento Teraspace" na página 50 para obter mais informações sobre o modelo de armazenamento teraspace.

| Para módulos, programas e programas de serviço, você tem a opção de especificar um dos seguintes | modelos de armazenamento para alguns compiladores: v Armazenamento de nível único (\*SNGLVL) v Teraspace (\*TERASPACE) v Inherit (\*INHERIT)

Este tópico discute o modelo de armazenamento teraspace.

**Especificando o modelo de armazenamento Teraspace**

| Para escolher o modelo de armazenamento teraspace para o seu programa RPG, COBOL, C ou C++, especifique o seguinte | quando você compila seu código:

| 1. Para C e C++, especifique \*SIM no parâmetro TERASPACE ao criar seus módulos.

1. Especifique \*TERASPACE ou \*INHERIT no parâmetro Storage model (STGMDL) do comando create module para sua linguagem de programação ILE.
2. Especifique \*TERASPACE no parâmetro STGMDL do comando Create Program (CRTPGM) ou Create Service Program (CRTSRVPGM). Essa opção deve ser compatível com o modelo de armazenamento dos módulos que você vincula ao programa. Consulte "Regras para módulos de vinculação" na página 47 para obter detalhes.

| Você também pode especificar \*TERASPACE no parâmetro STGMDL do programa Create Bound C

| (CRTBNDC), Criar Programa C++ Acoplado (CRTBNDCPP), Criar Programa RPG Acoplado

| (CRTBNDRPG) e os comandos Criar Programa COBOL Vinculado (CRTBNDCBL), que são criados em um | etapa um programa acoplado que contém apenas um módulo.

Nos comandos CRTPGM e CRTSRVPGM, você também pode especificar \*INHERIT no parâmetro STGMDL. Isso faz com que o programa ou programa de serviço seja criado de tal forma que ele possa usar armazenamento de nível único ou teraspace, dependendo do tipo de armazenamento em uso no grupo de ativação no qual o programa ou programa de serviço é ativado.

O uso do atributo \*INHERIT fornece a maior flexibilidade, mas você também deve especificar \*CALLER no parâmetro ATTIGRP. Nesse caso, lembre-se de que seu programa ou programa de serviço pode ser ativado com armazenamento de nível único ou teraspace, e você deve cuidar para que seu código possa lidar efetivamente com ambas as situações. Por exemplo, o tamanho total de todas as variáveis estáticas não deve ser maior do que os limites menores impostos para armazenamento de nível único.

*Tabela 3. Modelo de armazenamento permitido para tipos específicos de programas.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Modelo de armazenamento de programas** |  | **Tipo de programa** |  |
| **OPM \*PGM** | **ILE \*PGM** | **ILE \*SRVPGM** |
| \*TERASPACE | Não | Sim | Sim |
| \*HERDAR | Não | Sim, mas apenas com  ACTGRP(\*CALLER) | Sim, mas apenas com  ACTGRP(\*CALLER) |
| \*SNGLVL | Sim | Sim | Sim |

**Selecionando um grupo de ativação compatível**

Um grupo de ativação reflete o modelo de armazenamento do programa raiz que causou a criação do grupo de ativação. O modelo de armazenamento determina o tipo de armazenamento automático, estático e constante fornecido ao programa.

Os programas de modelo de armazenamento de nível único recebem armazenamento automático, estático e constante de nível único. Por padrão, esses programas também usarão armazenamento de nível único para armazenamento de pilha.

Os programas de modelo de armazenamento Teraspace recebem armazenamento automático, estático e constante do teraspace. Por padrão, esses programas também usarão teraspace para armazenamento de pilha.

| Os programas que usam o modelo de armazenamento teraspace não podem ser ativados em um grupo de ativação cuja raiz

| programa usa o modelo de armazenamento de nível único (isso inclui os grupos de ativação padrão do OPM). Programas | que usam o modelo de armazenamento de nível único não podem ser ativados em um grupo de ativação cujo programa raiz | usa o modelo de armazenamento teraspace.

A tabela a seguir resume a relação entre os modelos de armazenamento e o tipo de grupo de ativação.

| *Tabela 4. Relação dos modelos de armazenamento com os grupos de ativação*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Modelo de armazenamento de programas** |  | **Atributo de grupo de ativação** | |  |
| **\*CHAMADOR** | **\*DFTACTGRP** | **\*NOVO** | **Nomeado** |
| \*TERASPACE | Sim | Não permitido. | Sim | Sim |
| \*HERDAR | Sim | Não permitido. | Não permitido. | Não permitido. |
| \*SNGLVL | Sim | Sim | Sim | Sim |

|

|

|

|

|

|

|

Ao escolher o grupo de ativação no qual o programa ou programa de serviço é executado, considere as seguintes diretrizes: v Se o programa de serviço especificar STGMDL(\*INHERIT), você deverá especificar ACTGRP(\*CALLER). v Se o seu programa especifica STGMDL(\*TERASPACE):

* Especifique ACTGRP(\*NEW) ou um grupo de ativação nomeado.
* Especifique ACTGRP(\*CALLER) somente se você puder garantir que cada programa que chama seu programa use o modelo de armazenamento teraspace.

**Como os modelos de armazenamento interagem**

A consistência é necessária entre os módulos e programas que usam um modelo de armazenamento. Aqui estão as regras para garantir que os programas interajam corretamente.

v "Regras para módulos de ligação"

v "Regras para Vinculação a Programas de Serviço" na página 48 v "Regras para Ativação de Programas e Programas de Serviço" na página 48 v "Regras para Chamadas de Programas e Procedimentos" na página 48

#### Regras para módulos de vinculação

A tabela a seguir mostra as regras para módulos de vinculação:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Regras de vinculação: Vinculação do módulo M em um programa com um modelo de armazenamento especificado.** | | **O modelo de armazenamento do programa que está sendo criado.** | | |
| **Teraespaço** | **Herdar** | **Armazenamento de nível único** |
| **M** | **Teraespaço** | Teraespaço | Erro | Erro |
| **Herdar** | Teraespaço | Herdar | Armazenamento de nível único |
| **Armazenamento de nível único** | Erro | Erro | Armazenamento de nível único |

#### Regras para vinculação a programas de serviço

A tabela a seguir mostra as regras para vincular programas a programas de serviço de destino.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Regras de vinculação de programa de serviço: o programa de chamada ou o programa de serviço podem se vincular a um programa de serviço de destino?** | |  | **Modelo de armazenamento do programa de serviço de destino** | | |
| **Teraespaço** |  | **Herdar** | **Armazenamento de nível único** |
| **Modelo de armazenamento do programa de chamada ou programa de serviço** | **Teraespaço** | Sim |  | Sim | Sim1 |
| **Herdar**1 | Sim2 |  | Sim | Sim2 |
| **Armazenamento de nível único** | Sim1 |  | Sim | Sim |

**Anotações:**

1. O programa de serviço de destino deve ser executado em um grupo de ativação distinto. Por exemplo, o programa de serviço de destino não pode ter o atributo ACTGRP(\*CALLER). Não é possível misturar modelos de armazenamento em um único grupo de ativação.
2. Se o programa de chamada ou o programa de serviço usar o modelo de armazenamento herdado e o programa de serviço de destino usar armazenamento de nível único ou o modelo de armazenamento teraspace, você deverá garantir que o grupo de ativação no qual o programa de serviço de destino está ativado tenha o mesmo modelo de armazenamento que o programa de serviço de destino. Aqui está um exemplo: o programa de serviço A é criado com o modelo de armazenamento herdado. O programa de serviço B é criado com o modelo de armazenamento teraspace e tem o atributo de grupo de ativação \*CALLER. O programa de serviço A está vinculado ao programa de serviço B. Nesse caso, o programa de serviço A deve sempre ser ativado em um grupo de ativação com o modelo de armazenamento teraspace.

#### Regras para ativar programas e programas de serviço

Um programa de serviço que especifica o modelo de armazenamento herdado pode ser ativado em um grupo de ativação que executa programas que usam modelos de armazenamento de nível único ou de armazenamento teraspace. Caso contrário, o modelo de armazenamento do programa de serviço deve corresponder ao modelo de armazenamento de outros programas executados no grupo de ativação.

#### Regras para chamadas de programas e procedimentos

Programas e programas de serviço que usam modelos de armazenamento diferentes podem interoperar. Eles podem ser unidos e compartilhar dados, desde que estejam em conformidade com as regras e restrições descritas neste capítulo.

**Convertendo seu programa ou programa de serviço para herdar um modelo de armazenamento**

| Convertendo seus programas ou programas de serviço para herdar um modelo de armazenamento (especificando \*INHERIT no | STGMDL), você os habilita para uso em ambientes de armazenamento teraspace ou de nível único. | Certifique-se de que seu código antecipa e gerencia efetivamente ponteiros de e para teraspace e | armazenamento de nível único. Consulte "Usando o Teraspace: Práticas recomendadas" na página 51 para obter mais informações.

| Há duas maneiras de habilitar seus programas existentes ou programas de serviço para o armazenamento teraspace | modelo: você pode recriá-los ou, em algumas circunstâncias, alterar o modelo de armazenamento de | existentes Programas. Ao criar um programa, primeiro crie todos os seus módulos com o modelo de armazenamento \*INHERIT. | Em seguida, crie seu programa ILE ou programa de serviço com o modelo de armazenamento \*INHERIT.

| Como alternativa, você pode alterar o modelo de armazenamento de um programa existente ou programa de serviço de | armazenamento de nível único (\*SNGLVL) para \*INHERIT, usando o comando CHGPGM ou CHGSRVPGM, se o | a seguir são verdadeiros:

|  |  |  |
| --- | --- | --- |
| |  |  |  |  |  | | 1.  2.  3.  4.  5.  6. | O objeto é um programa ILE ou um programa de serviço.  O objeto usa o modelo de armazenamento de nível único.  O objeto usa o grupo de ativação de seu chamador.  O objeto é um programa cuja versão de destino é V6R1M0 ou posterior.  O objeto é um programa de serviço cuja versão de destino é V5R1M0 ou posterior.  Todos os módulos acoplados do objeto têm uma versão de destino de V5R1M0 ou posterior. |

**Atualizando seus programas: considerações sobre o Teraspace**

Você pode adicionar e substituir módulos dentro de um programa, desde que eles usem o mesmo modelo de armazenamento. No entanto, você não pode usar os comandos de atualização para alterar o modelo de armazenamento do módulo acoplado ou do programa.

**Aproveitando os ponteiros de 8 bytes em seu código C e C++**

Um ponteiro de 8 bytes pode apontar apenas para teraspace. Um ponteiro de procedimento de 8 bytes refere-se a um procedimento ativo através do teraspace. Os únicos tipos de ponteiros de 8 bytes são ponteiros de espaço e de procedimento.

Em contraste, existem muitos tipos de ponteiros de 16 bytes. A tabela a seguir mostra como os ponteiros de 8 e 16 bytes se comparam.

*Tabela 5. Comparação de ponteiros*

|  |  |  |
| --- | --- | --- |
| **Propriedade** | **Ponteiro de 8 bytes** | **Ponteiro de 16 bytes** |
| Comprimento (memória necessária) | 8 bytes | 16 bytes |
| Etiquetado | Não | Sim |
| Alinhamento | O alinhamento de bytes é permitido (ou seja, uma estrutura compactada). O alinhamento "natural" (8 bytes) é preferido para o desempenho. | Sempre de 16 bytes. |
| Atomicidade | Operações de carregamento atômico e armazenamento quando 8 bytes alinhados. Não se aplica a operações de cópia agregada. | Carga atômica e operações de armazenamento. Cópia atômica quando parte de um agregado. |
| Gama endereçável | Armazenamento Teraspace | Armazenamento Teraspace + armazenamento de nível único |
| Conteúdo do ponteiro | Um valor de 64 bits que representa um deslocamento para teraspace. Não contém um endereço eficaz. | Bits de tipo de ponteiro de 16 bytes e um endereço efetivo de 64 bits. |
| Localidade de referência | Processar referência de armazenamento local. (Um ponteiro de 8 bytes só pode fazer referência ao teraespaço do trabalho no qual a referência de armazenamento ocorre.) | Processar referência de armazenamento local ou de nível único. (Um ponteiro de 16 bytes pode fazer referência a um armazenamento que é logicamente de propriedade de outro trabalho.) |
| Operações permitidas | Operações específicas de ponteiro permitidas para ponteiros de espaço e ponteiros de procedimento, e usando uma exibição que não seja de ponteiro, todas as operações aritméticas e lógicas apropriadas a dados binários podem ser usadas sem invalidar o ponteiro. | Somente operações específicas de ponteiro. |
| Referências de armazenamento mais rápidas | Não | Sim |
| Cargas, armazenamentos e aritmética do ponteiro de espaço mais rápidos | Sim, incluindo evitar despesas gerais do EAO. | Não |
| Tamanho do valor binário preservado quando convertido no ponteiro | 8 bytes | 4 bytes |
| Pode ser aceito como um parâmetro por um procedimento que é um manipulador de exceção ou um manipulador de cancelamento. | Não | Sim |

**Suporte a ponteiro em compiladores C e C++**

Para aproveitar ao máximo os ponteiros de 8 bytes ao compilar seu código com o compilador IBM C ou C++, especifique STGMDL(\*TERASPACE) e DTAMDL(\*LLP64).

Os compiladores C e C++ também fornecem o seguinte suporte de ponteiro:

v Sintaxe para declarar explicitamente ponteiros de 8 ou 16 bytes:

* Declare um ponteiro de 8 bytes como char \* \_\_ptr64
* Declare um ponteiro de 16 bytes como char \* \_\_ptr128 opção de compilador v A e pragma para especificar o *modelo de dados*, que é exclusivo para o ambiente de programação C e C++. O modelo de dados afeta o tamanho padrão dos ponteiros na ausência de um dos qualificadores explícitos. Você tem duas opções para o modelo de dados:
* P128, também conhecido como 4-4-16[[1]](#footnote-1)
* LLP64, também conhecido como 4-4-8[[2]](#footnote-2)

**Conversões de ponteiro**

Os compiladores IBM C e C++ convertem \_\_ptr128 em \_\_ptr64 e vice-versa, conforme necessário, com base em declarações de função e variável. No entanto, as interfaces com parâmetros de ponteiro para ponteiro exigem tratamento especial.

Os compiladores inserem automaticamente conversões de ponteiro em comprimentos de ponteiro de correspondência. Por exemplo, as conversões são inseridas quando os argumentos de ponteiro para uma função não correspondem ao comprimento dos parâmetros de ponteiro no protótipo da função. Ou, se ponteiros de comprimentos diferentes forem comparados, o compilador converterá implicitamente o ponteiro de 8 bytes em um ponteiro de 16 bytes para a comparação. Os compiladores também permitem que conversões explícitas sejam especificadas, como conversões. Tenha estes pontos em mente se adicionar conversões de ponteiro:

v Uma conversão de um ponteiro de 16 bytes para um ponteiro de 8 bytes só funciona se o ponteiro de 16 bytes contiver um endereço teraspace ou um valor de ponteiro nulo. Caso contrário, uma exceção MCH0609 é sinalizada.

v ponteiros de 16 bytes não podem ter tipos convertidos de um para outro, mas um ponteiro OPEN de 16 bytes pode conter qualquer tipo de ponteiro. Em contraste, não existe nenhum ponteiro OPEN de 8 bytes, mas ponteiros de 8 bytes podem ser logicamente convertidos entre um ponteiro de espaço e um ponteiro de procedimento. Mesmo assim, uma conversão de ponteiro de 8 bytes é apenas uma exibição do tipo de ponteiro, portanto, não permite que um ponteiro de espaço seja realmente usado como um ponteiro de procedimento, a menos que o ponteiro de espaço tenha sido definido para apontar para um procedimento.

Ao adicionar conversões explícitas entre ponteiros e valores binários, lembre-se de que os ponteiros de 8 bytes e 16 bytes se comportam de maneira diferente. Um ponteiro de 8 bytes pode reter um valor binário completo de 8 bytes, enquanto um ponteiro de 16 bytes só pode reter um valor binário de 4 bytes. Ao manter um valor binário, a única operação definida para um ponteiro é uma conversão de volta para um campo binário. Todas as outras operações são indefinidas, incluindo o uso como um ponteiro, conversão para um comprimento de ponteiro diferente e comparação de ponteiro. Assim, por exemplo, se o mesmo valor inteiro fosse atribuído a um ponteiro de 8 bytes e a um ponteiro de 16 bytes, o ponteiro de 8 bytes fosse convertido em um ponteiro de 16 bytes e uma comparação de ponteiro de 16 bytes fosse feita, o resultado da comparação seria indefinido e provavelmente não produziria um resultado igual.

As comparações de ponteiro de comprimento misto são definidas somente quando um ponteiro de 16 bytes contém um endereço teraspace e um ponteiro de 8 bytes também o faz (ou seja, o ponteiro de 8 bytes não contém um valor binário). Em seguida, é válido converter o ponteiro de 8 bytes em um ponteiro de 16 bytes e comparar os dois ponteiros de 16 bytes. Em todos os outros casos, os resultados da comparação são indefinidos. Assim, por exemplo, se um ponteiro de 16 bytes foi convertido em um ponteiro de 8 bytes e, em seguida, comparado com um ponteiro de 8 bytes, o resultado é indefinido.

**Usando o modelo de armazenamento Teraspace**

Em um ambiente teraspace ideal, todos os seus módulos, programas e programas de serviço usariam o modelo de armazenamento teraspace. Em um nível prático, no entanto, você precisará gerenciar um ambiente que combine módulos, programas e programas de serviço que usam os dois modelos de armazenamento.

Esta seção descreve as práticas que você pode implementar para avançar em direção a um ambiente teraespacial ideal. Esta seção também discute como você pode minimizar os problemas potenciais de um ambiente que mistura programas que usam armazenamento de nível único e teraspace.

**Usando o Teraspace: Práticas recomendadas**

v *Use apenas módulos de modelo de armazenamento teraspace*

Crie seus módulos de modo que eles usem o teraspace ou herdem o modelo de armazenamento. Os módulos de armazenamento de nível único não são adequados para um ambiente teraspace porque você não pode vinculá-los ao programa. Se você absolutamente tiver que usá-los (por exemplo, se você não tiver acesso ao código-fonte do módulo), consulte o cenário 9 em "Dicas de uso do Teraspace" na página 52. v *Vincular somente a programas de serviço que usam o teraspace ou herdam o modelo de armazenamento*

Seu programa de modelo de armazenamento teraspace pode se vincular a praticamente qualquer tipo de programa de serviço. No entanto, ele normalmente se vincula apenas a programas de serviço de modelo de armazenamento herdeiro ou teraspace. Se você controlar os programas de serviço, você deve criar todos os seus programas de serviço de modo que eles possam herdar o modelo de armazenamento do programa que os vincula. Em geral, os programas de serviço IBM são criados dessa maneira. Talvez seja necessário fazer o mesmo, especialmente se você planeja fornecer seus programas de serviço a programadores de terceiros. Consulte o cenário 10 em "Dicas de uso do Teraspace" na página 52 se você absolutamente precisar se vincular a um programa de serviço de armazenamento de nível único.

Se você seguiu as diretrizes descritas neste tópico, você pode usar teraspace em seus programas. No entanto, o uso do teraspace requer que você preste muita atenção às suas práticas de codificação, porque o armazenamento de nível único é usado por padrão. Os tópicos a seguir descrevem as coisas que você não pode fazer com o teraspace e algumas coisas que você não deve fazer. Em alguns casos, o sistema impede que você execute determinadas ações, mas em outros momentos você deve gerenciar possíveis interações de teraspace e armazenamento de nível único por conta própria.

v "Controles do sistema sobre programas teraespaciais quando eles são criados" v "Controles do sistema sobre programas teraespaciais quando eles são ativados"

**Observação:** Os programas de serviço que usam o modelo de armazenamento herdado também devem seguir essas práticas, pois podem ser ativados para usar o teraspace.

#### Controles do sistema sobre programas Teraspace quando eles são criados

Na maioria dos casos, o sistema impede que você execute qualquer uma das seguintes ações:

v Combinação de módulos de modelo de armazenamento de nível único e armazenamento teraspace no mesmo programa ou programa de serviço.

v Criando um programa de modelo de armazenamento teraspace ou um programa de serviço que também especifica um grupo de ativação padrão (ACTGRP(\*DFTACTGRP)).

v Vinculando um programa de armazenamento de nível único a um programa de serviço de modelo de armazenamento teraspace que também especifica um grupo de ativação de \*CALLER.

#### Controles do sistema sobre programas Teraspace quando eles são ativados

Em alguns casos, no momento da ativação, o sistema determinará que você criou seus programas e programas de serviço de tal forma que os programas de modelo de armazenamento de nível único e teraspace ou programas de serviço tentem ser ativados no mesmo grupo de ativação. O sistema enviará a exceção de violação de acesso de ativação e falhará na ativação.

**Interfaces do IBM i e Teraspace**

As interfaces que têm parâmetros de ponteiro normalmente esperam ponteiros marcados de 16 bytes (\_\_ptr128):

v Você pode chamar interfaces com apenas um único nível de ponteiro (por exemplo, void f(char\*p); ) usando diretamente ponteiros de 8 bytes (\_\_ptr64), uma vez que o compilador converterá o ponteiro conforme necessário. Certifique-se de usar os arquivos de cabeçalho do sistema.

v Interfaces com múltiplos níveis de ponteiros (por exemplo, void g(char\*\*p); ) normalmente exigem que você declare explicitamente um ponteiro de 16 bytes para o segundo nível. No entanto, as versões que aceitam ponteiros de 8 bytes são fornecidas para a maioria das interfaces de sistema desse tipo, para permitir que elas sejam chamadas diretamente do código que usa apenas ponteiros de 8 bytes. Essas interfaces são habilitadas por meio dos arquivos de cabeçalho padrão quando você seleciona a opção datamodel(LLP64).

**APIs vinculáveis para usar o teraspace:**

A IBM fornece APIs vinculáveis para alocar e descartar teraspace. [[3]](#footnote-3)v \_C\_TS\_malloc() aloca o armazenamento dentro de um teraespaço.

| v \_C\_TS\_malloc64() permite que uma alocação de armazenamento teraespacial seja especificada usando um valor de tamanho de 8 bytes.

v \_C\_TS\_free() libera uma alocação anterior de teraespaço. v \_C\_TS\_realloc() altera o tamanho de uma alocação teraespacial anterior. v \_C\_TS\_calloc() aloca o armazenamento dentro de um teraespaço e o define como 0.

malloc(), free(), calloc(), e realloc() alocam ou desalocam armazenamento de nível único ou armazenamento teraespacial

de acordo com o modelo de armazenamento de seu programa de chamada, a menos que tenha sido compilado com a opção de compilador TERASPACE(\*YES \*TSIFC).

A memória compartilhada POSIX e as interfaces de arquivo mapeadas de memória podem usar teraspace. Para obter mais informações sobre APIs de Comunicação entre Processos e a interface shmget(), consulte o tópico *APIs do tipo UNIX*  no Centro de Informações do IBM i (no tópico Categoria de programação e API).

**Problemas potenciais que podem surgir quando você usa o Teraspace**

Quando você usa teraspace em seus programas, você deve estar ciente dos problemas potenciais que podem surgir.

| v *Se você criar programas usando o parâmetro TGTRLS(V5R4M0), tenha cuidado para não desenvolver uma dependência |*  passando endereços teraspace para outros programas *, a menos que todos esses outros programas possam lidar com endereços teraspace.*  Todos os | programas executados no IBM i 6.1 ou posterior podem usar endereços teraspace.

v *Algumas instruções MI não podem processar um endereço teraspace.*  Uma tentativa de usar um endereço teraspace nestas instruções causa uma exceção MCH0607.

* MATBPGM
* MATPG
* SCANX (apenas algumas opções são limitadas)

| v *Estouro de endereço efetivo (EAO) pode prejudicar o desempenho.*  Essa situação pode ocorrer em determinados códigos | gerado para um processador anterior ao POWER6®. Consulte "Adaptive Code Generation" na página 143.

| Especificamente, um EAO pode ocorrer durante um cálculo de endereço em um ponteiro de 16 bytes, quando o resultado é

| um valor de endereço teraspace menor que o limite de 16 MB menor que o endereço inicial. Um

| é gerada uma interrupção de hardware que é tratada pelo software do sistema. Muitas dessas interrupções podem

| afetam o desempenho. Evite cálculos frequentes de endereços teraespaciais que calculam um valor menor em um | área diferente de 16 MB. Ou, crie seu programa com o MinimizeTeraspaceFalseEAOs LICOPT, como | descrito em "Opções de código interno licenciado" na página 138.

**Dicas de uso do Teraspace**

Você pode encontrar os seguintes cenários ao trabalhar com o modelo de armazenamento teraspace. Soluções recomendadas são fornecidas.

v *Cenário 1: Você precisa de mais de 16 MB de armazenamento dinâmico em uma única alocação*

| Use \_C\_TS\_malloc ou crie seus programas para que o teraspace seja usado para armazenamento de pilha, conforme descrito | em "Usando o Teraspace para armazenamento" na página 45. v *Cenário 2: Você precisa de mais de 16 MB de memória compartilhada*

Use memória compartilhada (shmget) com a opção teraspace. v *Cenário 3: Você precisa acessar arquivos de fluxo de bytes grandes com eficiência* Use arquivos mapeados de memória (mmap).

| Você pode acessar arquivos mapeados na memória de qualquer programa, mas para obter o melhor desempenho, use o teraspace | modelo de armazenamento e, se o seu idioma o suportar, o modelo de dados de ponteiro de 8 bytes. v *Cenário 4: Você precisa de mais de 16 MB de armazenamento automático ou estático contíguo* | Use o modelo de armazenamento teraspace.

v *Cenário 5: Seu aplicativo faz uso pesado de ponteiros de espaço*

| Use o modelo de armazenamento teraspace e uma linguagem que ofereça suporte ao modelo de dados de ponteiro de 8 bytes para reduzir | espaço ocupado pela memória e acelere as operações de ponteiro.

v *Cenário 6: Você precisa portar o código de outro sistema e deseja evitar problemas que são exclusivos do uso do ponteiro de 16 bytes*

| Use o modelo de armazenamento teraspace e uma linguagem que ofereça suporte ao modelo de dados de ponteiro de 8 bytes.

v *Cenário 7: Você precisa usar o armazenamento de nível único em seu programa teraspace*

Às vezes, sua única opção é usar o armazenamento de nível único em seus programas de modelo de armazenamento teraspace. Por exemplo, talvez você precise dele para armazenar dados do usuário para comunicação entre processos. Você pode obter armazenamento de nível único de qualquer uma das seguintes fontes:

* Armazenamento em um espaço de usuário, obtido da API QUSCRTUS ou da instrução CRTS MI | – Interfaces de pilha de armazenamento de nível único em sua linguagem de programação
* Referência de armazenamento de nível único que foi passada para o programa
* Espaço de pilha de armazenamento de nível único obtido da instrução ALCHS MI v *Cenário 8: Aproveite os ponteiros de 8 bytes em seu código*

Crie seu módulo e programa com STGMDL(\*TERASPACE). Use DTAMDL(\*LLP64) ou declarações explícitas (\_\_ptr64) para obter ponteiros de 8 bytes para se referir ao teraespaço (em oposição aos ponteiros de 16 bytes apontando para o teraespaço). Em seguida, você obterá as vantagens listadas em "Aproveitando os ponteiros de 8 bytes em seu código C e C++" na página 49.

v *Cenário 9: Incorporando um módulo de modelo de armazenamento de nível único*

Não é possível vincular um módulo de armazenamento de nível único a um módulo de modelo de armazenamento teraspace. Se você precisar fazer isso, primeiro tente obter uma versão do módulo que use (ou herde) o modelo de armazenamento teraspace e, em seguida, simplesmente use-o conforme descrito em "Usando o Teraspace: Melhores Práticas" na página 51. Caso contrário, você tem duas opções:

* Empacote o módulo em um programa de serviço separado. O programa de serviço usará o modelo de armazenamento de nível único, portanto, use a abordagem fornecida no cenário 10, abaixo, para chamá-lo.
* Empacote o módulo em um programa separado. Este programa usará o modelo de armazenamento de nível único. Use a abordagem descrita no cenário 11, abaixo, para chamá-lo.

v *Cenário 10: Vinculação a um programa de serviço de modelo de armazenamento de nível único*

Você pode vincular seu programa teraspace a um programa de serviço que usa armazenamento de nível único se os dois programas de serviço forem ativados em grupos de ativação separados. Não é possível fazer isso se o programa de serviço de armazenamento de nível único especificar a opção ACTGRP(\*CALLER).

v *Cenário 11: Chamando funções que têm parâmetros de ponteiro para ponteiro*

Chamadas para algumas funções que têm parâmetros de ponteiro para ponteiro requerem manipulação especial de módulos compilados com a opção DTMDL(\*LLP64). Conversões implícitas entre ponteiros de 8 e 16 bytes se aplicam a parâmetros de ponteiro. Eles não se aplicam ao objeto de dados apontado pelo parâmetro de ponteiro, mesmo que esse destino de ponteiro também seja um ponteiro. Por exemplo, o uso de uma interface **char\*\*** declarada em um arquivo de cabeçalho que afirma o modelo de dados P128 comumente usado exigirá algum código em módulos criados com o modelo de dados LLP64. Certifique-se de passar o endereço de um ponteiro de 16 bytes para este caso. Aqui estão alguns exemplos:

* Neste exemplo, você criou um programa de modelo de armazenamento teraspace usando ponteiros de 8 bytes com o

Opções STGMDL (\*TERASPACE)DTAMDL(\*LLP64) em um comando create, como CRTCMOD. Agora você quer

para passar um ponteiro para um ponteiro para um caractere em uma matriz do seu programa de modelo de armazenamento teraspace para uma interface **char\*\***  P128. Para fazer isso, você deve declarar explicitamente um ponteiro de 16 bytes:

#pragma datamodel(P128) void func(char \*\*); #pragma datamodel(pop)

char myArray[32]; char \*\_ptr128 myPtr;

myPtr = myArray; /\* atribuir endereço da matriz ao ponteiro de 16 bytes \*/ func(&myPtr); /\* passar o endereço do ponteiro de 16 bytes para a função \*/

* Uma interface de programação de aplicativos (API) comumente usada com parâmetros de ponteiro para ponteiro é iconv. Ele espera apenas ponteiros de 16 bytes. Aqui está parte do arquivo de cabeçalho para **iconv**:

...

#pragma datamodel(P128) ...

size\_t inconv(iconv\_t cd,

char \*\*inbuf,

size\_t \*inbytesleft, char \*\*outbuf,

size\_t \*outbytesleft); ...

#pragma datamodel(pop) ...

O código a seguir chama **iconv** de um programa compilado com a opção DTAMDL(\*LLP64):

...

iconv\_t myCd; size\_t myResult;

char \*\_ptr128 myInBuf, myOutBuf; size\_t myInLeft, myOutLeft; ...

myResult = inconv(myCd, &myInBuf, &myInLeft, &myOutBuf, &myOutLeft); ...

Você também deve estar ciente de que o arquivo de cabeçalho da interface Retrieve Pointer to User Space (QUSPTRUS) especifica um parâmetro void\* onde um ponteiro para um ponteiro é realmente esperado. Certifique-se de passar o endereço de um ponteiro de 16 bytes para o segundo operando.

v *Cenário 12: Redeclaring functions*

Evite redeclarar funções que já estão declaradas em arquivos de cabeçalho fornecidos pela IBM. Normalmente, as declarações locais não têm os comprimentos de ponteiro corretos especificados. Uma dessas interfaces comumente usadas é **errno**, que é implementada como uma chamada de função no IBM i.

v *Cenário 13: Usando o modelo de dados \*LLP64 com programas e funções que retornam um ponteiro*

Se você estiver usando o modelo de dados \*LLP64, observe cuidadosamente os programas e funções que retornam um ponteiro. Se o ponteiro apontar para armazenamento de nível único, seu valor não poderá ser atribuído corretamente a um ponteiro de 8 bytes, portanto, os clientes dessas interfaces deverão manter o valor retornado em um ponteiro de 16 bytes. Uma dessas APIs é o QUSPTRUS. Os espaços do usuário residem no armazenamento de nível único.

**Capítulo 5. Conceitos de Criação de Programas**

O processo de criação de programas ILE ou programas de serviço oferece maior flexibilidade e controle na concepção e manutenção de aplicativos. O processo inclui duas etapas:

1. Compilando o código-fonte em módulos.
2. Vinculando módulos em um programa ILE ou programa de serviço. A associação ocorre quando o comando Criar Programa (CRTPGM) ou Criar Programa de Serviço (CRTSRVPGM) é executado.

Este capítulo explica os conceitos associados ao fichário e ao processo de criação de programas ILE ou programas de serviço. Antes de ler este capítulo, você deve estar familiarizado com os conceitos de vinculação descritos no Capítulo 2, "Conceitos Básicos da ILE", na página 9.

**Criar programa e criar comandos de programa de serviço**

Os comandos Criar Programa (CRTPGM) e Criar Programa de Serviço (CRTSRVPGM) são semelhantes e compartilham muitos dos mesmos parâmetros. Comparar os parâmetros usados nos dois comandos ajuda a esclarecer como cada comando pode ser usado.

A Tabela 6 mostra os comandos e seus parâmetros com os valores padrão fornecidos.

*Tabela 6. Parâmetros para comandos CRTPGM e CRTSRVPGM*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Grupo de parâmetros** | **Comando CRTPGM** | **Comando CRTSRVPGM** |
|  | Identificação | MÓDULO PGM(\*CURLIB/WORK)(\*PGM) | MÓDULO SRVPGM(\*CURLIB/UTILITÁRIO)(\*SRVPGM) |
|  | Acesso ao programa | ENTMOD(\*PRIMEIRO) | EXPORTAR(\*SRCFILE) SRCFILE(\*LIBL/ QSRVSRC) SRCMBR(\*SRVPGM) |
|  | Ligação | BNDSRVPGM(\*NENHUM) BNDDIR(\*NENHUM) | BNDSRVPGM(\*NENHUM) BNDDIR(\*NENHUM) |
|  | Tempo de execução | ACTGRP(\*ENTMOD) | ACTGRP(\*CALLER) |
|  | Optimização | IPA(\*NÃO) IPACTLFILE(\*NENHUM) ARGOPT(\*NÃO) | IPA(\*NÃO) IPACTLFILE(\*NENHUM) ARGOPT(\*NÃO) |
| |  |  |  |  |  |  |  | | Variado | OPÇÃO(\*GEN \*NODUPPROC  \*NODUPVAR \*AVISAR \*RSLVREF)  DETALHE(\*NENHUM) ALWUPD(\*SIM)  ALWLIBUPD(\*NÃO) ALWRINZ(\*NÃO)  SUBSTITUIR(\*SIM) AUT(\*LIBCRTAUT)  TEXTO(\*ENTMODTXT)  TGTRLS(\*ATUAL) USRPRF(\*USUÁRIO)  STGMDL(\*SNGLVL) | OPÇÃO(\*GEN \*NODUPPROC  \*NODUPVAR \*AVISAR \*RSLVREF)  DETALHE(\*NENHUM) ALWUPD(\*SIM)  ALWLIBUPD(\*NÃO) ALWRINZ(\*NÃO)  SUBSTITUIR(\*SIM) AUT(\*LIBCRTAUT)  TEXTO(\*ENTMODTXT)  TGTRLS(\*ATUAL) USRPRF(\*USUÁRIO)  STGMDL(\*SNGLVL) |

Os parâmetros de identificação para ambos os comandos nomeiam o objeto a ser criado e os módulos copiados.

A única diferença nos dois parâmetros está no nome do módulo padrão a ser usado ao criar o objeto.

Para CRTPGM, use o mesmo nome para o módulo conforme especificado no parâmetro program (\*PGM). Para CRTSRVPGM, use o mesmo nome para o módulo especificado no parâmetro do programa de serviço (\*SRVPGM). Caso contrário, esses parâmetros parecem e agem da mesma forma.

A semelhança mais significativa nos dois comandos é como o fichário resolve símbolos entre as importações e exportações. Em ambos os casos, o fichário processa a entrada dos parâmetros module (MODULE), bound service program (BNDSRVPGM) e binding directory (BNDDIR).

A diferença mais significativa nos comandos é com os parâmetros de acesso ao programa (consulte "Acesso ao programa" na página 64). Para o comando CRTPGM, tudo o que precisa ser identificado para o fichário é qual módulo tem o procedimento de entrada do programa. Uma vez que o programa é criado e uma chamada de programa dinâmico é

© Direitos autorais IBM Corp. 1997, 2010

feito para este programa, o processamento começa com o módulo que contém o procedimento de entrada do programa. O comando CRTSRVPGM precisa de mais informações de acesso ao programa porque ele pode fornecer uma interface de vários pontos de acesso para outros programas ou programas de serviço.

**Usar Autoridade Adotada (QUSEADPAUT)**

O valor do sistema QUSEADPAUT define quais usuários podem criar programas com o atributo de autoridade adotada de uso (USEADPAUT(\*YES)). Todos os usuários autorizados pelo valor do sistema QUSEADPAUT podem criar ou alterar programas e programas de serviço para usar a autoridade adotada, se tiverem as autoridades necessárias. Consulte Referência de segurança para descobrir quais autoridades são necessárias.

O valor do sistema pode conter o nome de uma lista de autorização. A autoridade do usuário é verificada em relação a essa lista. Se o usuário tiver pelo menos \*USE autoridade para a lista de autorização nomeada, o usuário poderá criar, alterar ou atualizar programas ou programas de serviço com o atributo USEADPAUT(\*YES). A autoridade para a lista de autorização não pode vir da autoridade adotada.

Se uma lista de autorização for nomeada no valor do sistema e a lista de autorização estiver ausente, a função que está sendo tentada não será concluída. Uma mensagem é enviada indicando isso. No entanto, se o programa for criado com a API QPRCRTPG e o valor \*NOADPAUT for especificado no modelo de opção, o programa será criado com êxito, mesmo que a lista de autorização não exista. Se mais de uma função for solicitada no comando ou API e a lista de autorização estiver ausente, a função não será executada.

*Tabela 7. Valores possíveis para QUSEADPAUT*

|  |  |
| --- | --- |
| **Valores** | **Descrição** |
| *nome da lista de autorização* | Uma mensagem de diagnóstico é sinalizada para indicar que o programa é criado com USEADPAUT(\*NO) se todos os itens a seguir forem verdadeiros:  v Uma lista de autorização é especificada para o valor do sistema QUSEADPAUT.  v O usuário não tem autoridade para a lista de autorização mencionada acima. v Não há outros erros quando o programa ou programa de serviço é criado.  Se o usuário tiver autoridade para a lista de autorização, o programa ou programa de serviço será criado com USEADPAUT(\*YES). |

**\*NENHUM** Todos os usuários autorizados pelo valor do sistema QUSEADPAUT podem criar ou alterar

programas e programas de serviço para usar a autoridade adotada se os usuários tiverem as autoridades necessárias.

**Usando parâmetros de otimização**

Especifique parâmetros de otimização para otimizar ainda mais seus programas vinculados a ILE ou programas de serviço. Para obter mais informações sobre parâmetros de otimização a serem usados ao criar ou modificar programas, consulte o Capítulo 13, "Técnicas avançadas de otimização", na página 125.

**Dados armazenados em módulos e programas**

Quando um módulo é criado, ele contém dados úteis, além do código compilado. Dependendo do compilador que foi usado, os módulos podem conter os seguintes tipos de dados:

v Dados de Criação (\*CRTDTA)

Isso é necessário para recriar ou converter o módulo (por exemplo, para alterar seu nível de otimização). Os módulos recém-criados sempre contêm dados de criação.

v Dados de depuração (\*DBGDTA)

Isso é necessário para depurar o programa ao qual o módulo está vinculado. Consulte o Capítulo 10, "Considerações sobre depuração", na página 113 e "Análise interprocessual (IPA)" na página 130 para obter mais informações sobre depuração.

v Dados de linguagem intermediária (\*ILDTA)

Isso é necessário para a técnica avançada de otimização da análise interprocessual (IPA). Consulte o Capítulo 13, "Técnicas avançadas de otimização", na página 125 para obter mais informações.

Use o comando Display Module (DSPMOD) para descobrir que tipos de dados são armazenados em um módulo.

O processo de associação copia dados do módulo para o programa ou programa de serviço que é criado. Os Dados de Idioma Intermediário (\*ILDTA) não são copiados para programas ou programas de serviço.

Use os comandos Programa de Exibição (DSPPGM) ou Programa de Serviço de Exibição (DSPSRVPGM) com o parâmetro DETAIL(\*MODULE) para ver os detalhes. Os programas podem conter os seguintes tipos de dados, além dos dados do módulo:

v Dados de Criação (\*CRTDTA)

Isso é necessário para recriar o programa ou programa de serviço. Os programas recém-criados sempre contêm esses dados de criação.

v Dados de criação de perfil de pedido de bloco (\*BLKORD)

Isso é gerado para criação de perfil de aplicativo. Consulte o Capítulo 13, "Técnicas avançadas de otimização", na página 125 para obter mais informações.

v Dados de criação de perfil de ordem de procedimento (\*PRCORD)

Isso é gerado para criação de perfil de aplicativo. Consulte o Capítulo 13, "Técnicas avançadas de otimização", na página 125 para obter mais informações.

Dados de Criação (\*CRTDTA) podem existir para o próprio programa e para cada módulo acoplado. Quando os dados são criados e armazenados com o módulo ou programa, eles são observáveis. O sistema operacional pode usar os dados para operações, como executar os comandos Change Module (CHGMOD) e Change Program (CHGPGM), depurar o programa e usar o IPA.

Você pode remover a observabilidade com o parâmetro Remover informações observáveis (RMVOBS) no

Módulo de Mudança (CHGMOD), Programa de Mudança (CHGPGM) e Programa de Serviço de Mudança

(CHGSRVPGM) comandos. Se você remover a observabilidade, os programas de MI não poderão mais acessar os dados.

**Nota:** Não é possível desfazer a remoção da observabilidade.

Para a maioria dos tipos de dados, a remoção da observabilidade remove esses dados do objeto. Se você não precisar mais da função correspondente e quiser tornar seu objeto menor, poderá remover os dados. No entanto, os Dados de Criação (\*CRTDTA) não são removidos do objeto; ela é transformada em uma forma não observável. O sistema operacional não pode usar os dados, mas a máquina pode usar dados de criação não observáveis para converter o objeto.

**Resolução de símbolos**

**Resolução de símbolo** é o processo pelo qual o fichário passa para corresponder ao seguinte: v As solicitações de importação do conjunto de módulos a serem vinculados pela cópia v O conjunto de exportações fornecidas pelos módulos e programas de serviço especificados

O conjunto de exportações a serem usadas durante a resolução de símbolos pode ser considerado como uma lista ordenada (numerada sequencialmente). A ordem das exportações é determinada pelo seguinte:

v A ordem em que os objetos são especificados nos parâmetros MODULE, BNDSRVPGM e BNDDIR

do comando CRTPGM ou CRTSRVPGM

v As exportações das rotinas de tempo de execução da linguagem dos módulos especificados

**Importações resolvidas e não resolvidas**

Uma importação e exportação consistem em um procedimento ou tipo de dados e um nome. Uma **importação não resolvida** é aquela cujo tipo e nome ainda não correspondem ao tipo e ao nome de uma exportação. Uma **importação resolvida** é aquela cujo tipo e nome correspondem exatamente ao tipo e ao nome de uma exportação.

Somente as importações dos módulos que estão vinculados por cópia vão para a lista de importação não resolvida. Durante a resolução do símbolo, a próxima importação não resolvida é usada para pesquisar uma correspondência na lista ordenada de exportações. Se existir uma importação não resolvida após a verificação do conjunto de exportações ordenadas, o objeto de programa ou programa de serviço normalmente não será criado. No entanto, se \*UNRSLVREF for especificado no parâmetro option, um objeto de programa ou programa de serviço com importações não resolvidas poderá ser criado. Se esse objeto de programa ou programa de serviço tentar usar uma importação não resolvida em tempo de execução, ocorrerá o seguinte:

v Se o objeto de programa ou programa de serviço foi criado ou atualizado para um sistema Versão 2 Release 3, mensagem de erro MCH3203 é emitida. Essa mensagem diz: "Erro de função na instrução da máquina".

v Se o objeto de programa ou programa de serviço foi criado ou atualizado para um sistema Versão 3 Release 1 ou posterior, a mensagem de erro MCH4439 é emitida. Essa mensagem diz: "Tente usar uma importação que não foi resolvida".

**Vinculação por cópia**

Os módulos especificados no parâmetro MODULE são sempre vinculados por cópia. Os módulos nomeados em um diretório de vinculação especificado pelo parâmetro BNDDIR são vinculados por cópia se forem necessários. Um módulo nomeado em um diretório de associação é necessário em um dos seguintes casos:

v O módulo fornece uma exportação para uma importação não resolvida v O módulo fornece uma exportação nomeada no bloco de exportação atual do arquivo de origem do idioma do fichário que está sendo usado para criar um programa de serviço

Se uma exportação encontrada na linguagem do fichário vier de um objeto de módulo, esse módulo será sempre vinculado por cópia, independentemente de ter sido explicitamente fornecido na linha de comando ou vir de um diretório de vinculação. Por exemplo

Módulo M1: importações P2

Módulo M2: exportações P2

Módulo M3: exportações P3

Linguagem do fichário S1: STRPGMEXP PGMLVL(\*CURRENT)

EXPORTAÇÃO P3

ENDPGMEXP

Diretório de vinculação BNDDIR1: M2

M3 |

CRTSRVPGM SRVPGM(MYLIB/SRV1) MÓDULO(MYLIB/M1) SRCFILE(MYLIB/S1)

SRCMBR(S1) BNDDIR(MYLIB/BNDDIR1)

O programa de serviço SRV1 terá três módulos: M1, M2 e M3. M3 é copiado porque P3 está no bloco de exportação atual.

**Vinculação por Referência**

Os programas de serviço especificados no parâmetro BNDSRVPGM são vinculados por referência. Se um programa de serviço nomeado em um diretório de associação fornecer uma exportação para uma importação não resolvida, esse programa de serviço será vinculado por referência. Um programa de serviço vinculado dessa maneira não adiciona novas importações.

**Nota:** Para controlar melhor o que fica vinculado ao seu programa, especifique o nome genérico do programa de serviço ou bibliotecas específicas. O valor \*LIBL só deve ser especificado em um ambiente controlado pelo usuário quando você sabe exatamente o que está sendo vinculado ao seu programa. Não especifique BNDSRVPGM(\*LIBL/ \*ALL) com OPTION(\*DUPPROC \*DUPVAR). Especificar \*LIBL com \*ALL pode fornecer resultados imprevisíveis em tempo de execução do programa.

**Vinculando um grande número de módulos**

Para o parâmetro module (MODULE) nos comandos CRTPGM e CRTSRVPGM, há um limite no número de módulos que você pode especificar. Se o número de módulos que você deseja vincular exceder o limite, você pode usar um dos seguintes métodos:

1. Use diretórios de associação para vincular um grande número de módulos que fornecem exportações que são necessárias para outros módulos.
2. Use uma convenção de nomenclatura de módulo que permita que nomes de módulo genéricos sejam especificados no parâmetro MODULE nos comandos CRTPGM e CRTSRVPGM. Por exemplo, CRTPGM PGM(mylib/payroll) MODULE(mylib/pay\*). Todos os módulos com nomes que começam com o pagamento são incondicionalmente incluídos no programa mylib / folha de pagamento. Portanto, escolha sua convenção de nomenclatura com cuidado para que os nomes genéricos especificados nos comandos CRTPGM ou CRTSRVPGM não vinculem módulos indesejados.
3. Agrupe os módulos em bibliotecas separadas para que o valor \*ALL possa ser usado com nomes de biblioteca específicos no parâmetro MODULE. Por exemplo, CRTPGM PGM(mylib/payroll) MODULE(payroll/\*ALL). Cada módulo na folha de pagamento da biblioteca é incondicionalmente incluído no programa mylib / payroll.
4. Use uma combinação de nomes genéricos e bibliotecas específicas que são descritas nos métodos 2 e 3.
5. Para programas de serviço, use o idioma de origem de associação. Uma exportação especificada no idioma de origem da associação faz com que um módulo seja vinculado se ele satisfizer a exportação. O comando RTVBNDSRC pode ajudá-lo a criar seu idioma de origem de vinculação. Embora o parâmetro MODULE no comando RTVBNDSRC limite o número de módulos que podem ser explicitamente especificados no parâmetro MODULE, você pode usar nomes de módulo genéricos e o valor \*ALL com nomes de biblioteca específicos. Você pode usar o comando RTVBNDSRC várias vezes com a saída direcionada para o mesmo arquivo de origem. No entanto, talvez seja necessário editar o idioma de origem da associação nesse caso.

| **Símbolos duplicados**

| Os símbolos para nomes de variáveis e nomes de procedimentos podem ser exportados de módulos como fracos | exportações ou exportações fortes.

| Exportações fracas são suportadas por várias linguagens de programação, conforme descrito na seção "Importar e

| Export Concepts" na página 66. Quando um módulo define e faz referência a um símbolo fraco, o símbolo é ambos

| fracamente exportado e importado. Vários módulos podem definir o mesmo símbolo como uma exportação fraca. Durante | resolução de símbolo, a exportação de um módulo é escolhida para satisfazer todas as importações correspondentes, de modo que todos os | módulos referem-se ao mesmo item.

| Exportações fortes são geralmente definidas por um único módulo. Se mais de um módulo definir o símbolo como

| uma exportação forte, uma mensagem de erro é emitida e a criação do programa falhará. Sobre a criação do programa

| comandos (CRTPGM, CRTSRVPGM, UPDPGM, UPDSRVPGM), você pode usar OPTION(\*DUPPROC

| \* DUPVAR \*WARN) para permitir que o comando continue apesar do aviso de exportação de símbolo duplicado

| Mensagens. Essas mensagens de erro são importantes: quando um módulo se refere a um símbolo que ele exporta, o

| a referência é feita como uma referência local, não como uma referência de importação. Se dois módulos definirem o mesmo símbolo

| como uma exportação forte, e cada módulo refere-se ao símbolo, eles estão se referindo a itens separados. Cuidadosamente

| examine essas mensagens para garantir que seu aplicativo funcione corretamente. Talvez seja necessário alterar o

| exportações fortes para exportações fracas ou alterar o código para que apenas um único módulo exporte o símbolo e | todos os outros módulos o importam.

**Importância da Ordem das Exportações**

Com apenas uma pequena alteração no comando, você pode criar um programa diferente, mas potencialmente igualmente válido. A ordem em que os objetos são especificados nos parâmetros MODULE, BNDSRVPGM e BNDDIR geralmente é importante somente se ambos os itens a seguir forem verdadeiros:

v Vários módulos ou programas de serviço estão exportando nomes de símbolos duplicados v Outro módulo precisa importar o nome do símbolo

A maioria dos aplicativos não tem símbolos duplicados, e os programadores raramente precisam se preocupar com a ordem em que os objetos são especificados. Para os aplicativos que têm símbolos duplicados exportados que também são importados, considere a ordem em que os objetos são listados nos comandos CRTPGM ou CRTSRVPGM.

Os exemplos a seguir mostram como a resolução de símbolos funciona. Os módulos, programas de serviço e diretórios de associação na Figura 29 na página 60 são usados para as solicitações CRTPGM na Figura 30 na página 61 e na Figura 31 na página 63. Suponha que todas as exportações e importações identificadas sejam procedimentos.

Os exemplos também mostram o papel dos diretórios de vinculação no processo de criação do programa. Suponha que a biblioteca MYLIB está na lista de bibliotecas para os comandos CRTPGM e CRTSRVPGM. O comando a seguir cria o diretório de vinculação L na biblioteca MYLIB:

CRTBNDDIR BNDDIR (MYLIB/L)

O comando a seguir adiciona os nomes dos módulos M1 e M2 e dos programas de serviço S e T ao diretório de vinculação L:

|  |  |  |  |
| --- | --- | --- | --- |
| Lista de Importação   |  | | --- | | P20 | | Pág. 21 | | Impressões |   .  .  . |

ADDBNDDIRE BNDDIR(MYLIB/L) OBJ((M1 \*MODULE) (M2 \*MODULE) (S) (T)) **Módulo M1**

**Módulo M2**

Lista de Importação

Lista de exportação

Pág. 30

P20

.

.

.

.

.

.

**Programa de Serviço S**

**Programa de Serviço T**

Lista de exportação

P1

P20

Pág. 30

.

.

.

Lista de exportação

Pág. 30

P40

Pág. 21

.

.

.

**Programa de Serviço**

**QLEPRINTS**

**Diretório de vinculação L**

\*MÓDULO

\*MÓDULO

\*SRVPGM

\*SRVPGM

M1

| M2

S

T

\*LIBL

\*LIBL

\*LIBL

\*LIBL

Lista de exportação

Impressões

.

.

.

RV2W1054-3

*Figura 29. Módulos, programas de serviço e diretório de vinculação*

#### Exemplo 1 de criação de programa

Suponha que o seguinte comando é usado para criar o programa A na Figura 30 na página 61:

CRTPGM PGM (TESTE/A)

MÓDULO (\*LIBL/M1)

BNDSRVPGM(\*LIBL/S)

BNDDIR(\*LIBL/L)

OPÇÃO(\*DUPPROC)

**Programa de Serviço S**

**Programa de Serviço T**

**Programa de Serviço**

**QLEPRINTS**

**Diretório de vinculação L**

\*MÓDULO

\*MÓDULO

\*SRVPGM

\*SRVPGM

M1

| M2

S

T

\*LIBL

\*LIBL

\*LIBL

\*LIBL

Lista de exportação

P1

P20

Pág. 30

.

.

.

Lista de exportação

Pág. 30

P40

Pág. 21

.

.

.

Lista de exportação

Impressões

.

.

.

**Módulo M2**

**Módulo M1**

Lista de Importação

P20

Pág. 21

Impressões

.

.

.

Lista de Importação

Lista de exportação

Pág. 30

P20

.

.

.

.

.

.

**Programa A**

RV2W1049-4

*Figura 30. Resolução de símbolos e criação de programas: Exemplo 1*

Para criar o programa A, o fichário processa objetos especificados nos parâmetros de comando CRTPGM na ordem especificada:

1. O valor especificado no primeiro parâmetro (PGM) é A, que é o nome do programa a ser criado.
2. O valor especificado no segundo parâmetro (módulo) é M1. O fichário começa aí. O módulo M1 contém três importações que precisam ser resolvidas: P20, P21 e Impressões.
3. O valor especificado no terceiro parâmetro (BNDSRVPGM) é S. O fichário verifica a lista de exportação do programa de serviço S em busca de procedimentos que resolvam quaisquer solicitações de importação não resolvidas. Como a lista de exportação contém o procedimento P20, essa solicitação de importação é resolvida.
4. O valor especificado no quarto parâmetro (BNDDIR) é L. Em seguida, o fichário verifica o diretório de vinculação L.
   1. O primeiro objeto especificado no diretório de vinculação é o módulo M1. O módulo M1 é conhecido atualmente porque foi especificado no parâmetro module, mas não fornece exportações.
   2. O segundo objeto especificado no diretório de vinculação é o módulo M2. O módulo M2 fornece exportações, mas nenhuma delas corresponde a nenhuma solicitação de importação não resolvida no momento (P21 e Prints).
   3. O terceiro objeto especificado no diretório de vinculação é o programa de serviço S. O programa de serviço S já foi processado na etapa 3 e não fornece nenhuma exportação adicional.
   4. O quarto objeto especificado no diretório de vinculação é o programa de serviço T. O fichário verifica a lista de exportação do programa de serviço T. O procedimento P21 é encontrado, o que resolve essa solicitação de importação.
5. A importação final que precisa ser resolvida (Impressões) não é especificada em nenhum parâmetro. No entanto, o fichário localiza o procedimento Prints na lista de exportação do programa de serviço QLEPRINTS, que é uma rotina de tempo de execução comum fornecida pelo compilador neste exemplo. Ao compilar um módulo, o compilador especifica como padrão o diretório de vinculação que contém seus próprios programas de serviço de tempo de execução e os programas de serviço de tempo de execução ILE. É assim que o fichário sabe que deve procurar quaisquer referências não resolvidas restantes nos programas de serviço de tempo de execução fornecidos pelo compilador. Se, depois que o fichário procurar nos programas de serviço de tempo de execução, houver referências que não podem ser resolvidas, a associação normalmente falhará. No entanto, se você especificar OPTION(\*UNRSLVREF) no comando create, o programa será criado.

#### Exemplo 2 de criação de programa

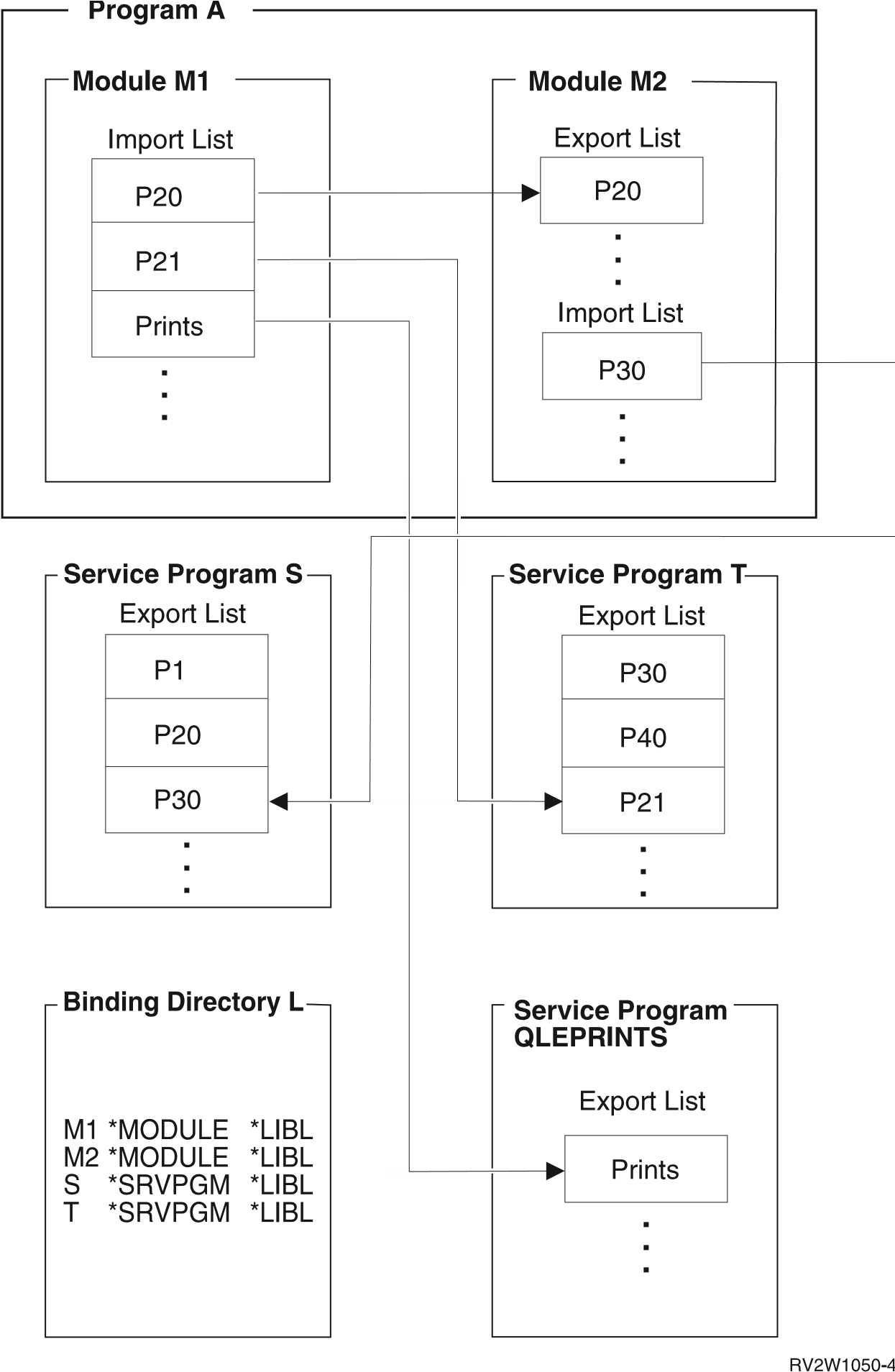
A Figura 31 na página 63 mostra o resultado de uma solicitação CRTPGM semelhante, exceto que o programa de serviço no parâmetro BNDSRVPGM foi removido:

CRTPGM PGM (TESTE/A)

MÓDULO (\*LIBL/M1)

BNDDIR(\*LIBL/L)

OPÇÃO(\*DUPPROC)



*Figura 31. Resolução de símbolos e criação de programas: Exemplo 2*

A alteração na ordenação dos objetos a serem processados altera a ordenação das exportações. Isso também resulta na criação de um programa que é diferente do programa criado no exemplo 1. Como o programa de serviço S não é especificado no parâmetro BNDSRVPGM do comando CRTPGM, o diretório de associação é processado. O módulo M2 exporta o procedimento P20 e é especificado no diretório de vinculação antes do programa de serviço S. Portanto, o módulo M2 é copiado para o objeto de programa resultante neste exemplo. Ao comparar a Figura 30 na página 61 com a Figura 31, você verá o seguinte:

v O programa A no exemplo 1 contém apenas o módulo M1 e usa procedimentos dos programas de serviço S, T e QLEPRINTS. v No programa A do exemplo 2, dois módulos chamados M1 e M2 usam os programas de serviço T e QLEPRINTS.

O programa no exemplo 2 é criado da seguinte maneira:

1. O primeiro parâmetro (PGM) especifica o nome do programa a ser criado.
2. O valor especificado no segundo parâmetro (MODULE) é M1, portanto, o fichário começa novamente lá. O módulo M1 contém as mesmas três importações que precisam ser resolvidas: P20, P21 e Impressões.
3. Desta vez, o terceiro parâmetro especificado não é BNDSRVPGM. É o BNDDIR. Portanto, o fichário primeiro verifica o diretório de vinculação especificado (L).
   1. A primeira entrada especificada no diretório de vinculação é o módulo M1. O módulo M1 desta biblioteca já foi processado pelo parâmetro module.
   2. A segunda entrada especificada no diretório de vinculação é para o módulo M2. O fichário verifica a lista de exportação do módulo M2. Como essa lista de exportação contém P20, essa solicitação de importação é resolvida. O módulo M2 é vinculado por cópia e suas importações devem ser adicionadas à lista de solicitações de importação não resolvidas para processamento. As solicitações de importação não resolvidas agora são P21, Prints e P30.
   3. O processamento continua para o próximo objeto especificado no diretório de vinculação, o programa de serviço 'S'. Aqui, o programa de serviço S fornece a exportação P30 para solicitações de importação atualmente não resolvidas de P21 e Impressões. O processamento continua para o próximo objeto listado no diretório de vinculação, o programa de serviço T.
   4. O programa de serviço T fornece exportação P21 para a importação não resolvida.
4. Como no exemplo 1, a solicitação de importação Prints não é especificada. No entanto, o procedimento é encontrado nas rotinas de tempo de execução fornecidas pela linguagem na qual o módulo M1 foi escrito.

A resolução do símbolo também é afetada pela força das exportações. Para obter informações sobre exportações fortes e fracas, consulte Exportar em "Conceitos de importação e exportação" na página 66.

**Acesso ao Programa**

Ao criar um objeto de programa ILE ou um objeto de programa de serviço, você precisa especificar como outros programas podem acessar esse programa. No comando CRTPGM, você faz isso com o parâmetro ENTMOD (módulo de entrada). No comando CRTSRVPGM, você faz isso com o parâmetro export (EXPORT) (consulte a Tabela 6 na página 55).

**Parâmetro do módulo de procedimento de entrada do programa no comando CRTPGM**

O parâmetro ENTMOD (módulo de procedimento de entrada de programa) informa ao fichário o nome do módulo no qual os itens a seguir estão localizados:

Procedimento de entrada no programa (PEP)

Procedimento de entrada do usuário (UEP)

Essas informações identificam qual módulo contém o PEP que obtém controle ao fazer uma chamada dinâmica para o programa que é criado.

O valor padrão para o parâmetro ENTMOD é \*FIRST. Esse valor especifica que o fichário usa como módulo de entrada o primeiro módulo que ele encontra na lista de módulos especificados no parâmetro module que contém um PEP.

Se existirem as seguintes condições:

\*FIRST é especificado no parâmetro ENTMOD Um segundo módulo com um PEP é encontrado o fichário copia este segundo módulo no objeto de programa e continua o processo de ligação. O fichário ignora o PEP adicional.

Se \*ONLY for especificado no parâmetro ENTMOD, somente um módulo no programa poderá conter um PEP. Se \*ONLY for especificado e um segundo módulo com um PEP for encontrado, o programa não será criado.

Para controle explícito, você pode especificar o nome do módulo que contém o PEP. Quaisquer outros PEPs são ignorados. Se o módulo explicitamente especificado não contiver um PEP, a solicitação CRTPGM falhará.

Para ver se um módulo tem um procedimento de entrada de programa, use o comando display module (DSPMOD). As informações aparecem no campo *Nome do procedimento de entrada* do programa da exibição Exibir informações do módulo. Se \*NONE for especificado no campo, este módulo não terá um PEP. Se um nome for especificado no campo, este módulo terá um PEP.

**Exportar parâmetro no comando CRTSRVPGM**

Os parâmetros export (EXPORT), source file (SRCFILE) e source member (SRCMBR) identificam a interface pública para o programa de serviço que está sendo criado. Os parâmetros especificam as exportações (procedimentos e dados) que um programa de serviço disponibiliza para uso por outros programas ILE ou programas de serviço.

O valor padrão para o parâmetro export é \*SRCFILE. Esse valor direciona o fichário para o parâmetro SRCFILE para uma referência a informações sobre exportações do programa de serviço. Essas informações adicionais são um arquivo de origem com a fonte do idioma do fichário (consulte "Binder Language" na página 67). O fichário localiza a origem do idioma do fichário e, a partir dos nomes especificados a serem exportados, gera uma ou mais assinaturas. A linguagem do fichário também permite que você especifique uma assinatura de sua escolha, em vez de fazer com que o fichário gere uma.

O comando Recover Binder Source (RTVBNDSRC) pode ser usado para criar um arquivo de origem que contenha a origem do idioma do fichário. A origem pode ser baseada em um programa de serviço existente ou em um conjunto de módulos. Se baseado em um programa de serviço, a origem é apropriada para recriar ou atualizar esse programa de serviço. Se baseado em um conjunto de módulos, a origem contém todos os símbolos elegíveis para serem exportados dos módulos. Em ambos os casos, você pode editar esse arquivo para incluir apenas os símbolos que deseja exportar e, em seguida, você pode especificar esse arquivo usando o parâmetro SRCFILE dos comandos CRTSRVPGM ou UPDSRVPGM.

O outro valor possível para o parâmetro export é \*ALL. Quando EXPORT(\*ALL) é especificado, todos os símbolos exportados dos módulos copiados são exportados do programa de serviço. A assinatura que é gerada é determinada pelo seguinte:

v O número de símbolos exportados v Ordem alfabética dos símbolos exportados

Se EXPORT(\*ALL) for especificado, nenhuma linguagem de fichário será necessária para definir as exportações de um programa de serviço. Ao especificar esse valor, você não precisa gerar a origem do idioma do fichário. No entanto, um programa de serviço com EXPORT(\*ALL) especificado pode ser difícil de atualizar ou corrigir se as exportações forem usadas por outros programas. Se o programa de serviço for alterado, a ordem ou o número de exportações poderá ser alterado. Portanto, a assinatura desse programa de serviço pode ser alterada. Se a assinatura for alterada, todos os programas ou programas de serviço que usam o programa de serviço alterado deverão ser recriados.

EXPORT(\*ALL) indica que todos os símbolos exportados dos módulos usados no programa de serviço são exportados do programa de serviço. A ILE C pode definir as exportações como globais ou estáticas. Somente as variáveis externas declaradas na ILE C como globais estão disponíveis com EXPORT(\*ALL). No ILE RPG, os seguintes itens estão disponíveis com EXPORT(\*ALL):

v O nome do procedimento principal do RPG v Os nomes de quaisquer subprocedimentos exportados v Variáveis definidas com a palavra-chave EXPORT

No ILE COBOL, os seguintes elementos de idioma são exportações de módulo:

v O nome no parágrafo PROGRAM-ID no programa COBOL lexicamente externo (não confundir com o objeto \*PGM) de uma unidade de compilação. Isso mapeia para uma exportação de procedimento forte.

v O nome gerado pelo compilador COBOL derivado do nome no parágrafo PROGRAM-ID no marcador anterior se esse programa não tiver o atributo INITIAL. Isso mapeia para uma exportação de procedimento forte. Para obter informações sobre exportações fortes e fracas, consulte Exportar em "Conceitos de importação e exportação" na página 66.

v Qualquer item de dados ou item de arquivo declarado como EXTERNO. Isso mapeia para uma exportação fraca.

#### Parâmetro de exportação usado com parâmetros de arquivo de origem e membro de origem

O valor padrão no parâmetro export é \*SRCFILE. Se \*SRCFILE for especificado no parâmetro export, o fichário também deverá usar os parâmetros SRCFILE e SRCMBR para localizar a origem do idioma do fichário.

O comando de exemplo a seguir vincula um programa de serviço chamado UTILITY usando os padrões para localizar a origem do idioma do fichário:

CRTSRVPGM SRVPGM(\*CURLIB/UTILITÁRIO)

MÓDULO(\*SRVPGM)

EXPORTAR(\*SRCFILE)

SRCFILE(\*LIBL/QSRVSRC)

SRCMBR(\*SRVPGM)

Para que este comando crie o programa de serviço, um membro chamado UTILITY deve estar no arquivo de origem QSRVSRC. Esse membro deve então conter a origem do idioma do fichário que o fichário traduz em uma assinatura e um conjunto de identificadores de exportação. O padrão é obter a origem do idioma do fichário de um membro com o mesmo nome que o nome do programa de serviço, UTILITY. Se uma fonte de idioma de arquivo, membro ou fichário com os valores fornecidos nesses parâmetros não estiver localizada, o programa de serviço não será criado.

#### Largura máxima de um arquivo para o parâmetro SRCFILE

Em versões V3R7 ou posteriores, a largura máxima de um arquivo para o parâmetro Arquivo de Origem (SRCFILE) no comando CRTSRVPGM ou UPDSRVPGM é de 240 caracteres. Se o arquivo for maior que a largura máxima, a mensagem CPF5D07 será exibida. Para V3R2, a largura máxima é de 80 caracteres. Para V3R6, V3R1 e V2R3, não há limite para a largura máxima.

**Conceitos de Importação e Exportação**

Os idiomas ILE oferecem suporte aos seguintes tipos de exportações e importações:

v Exportações de dados fracas v Importações de dados fracas v Exportações de dados fortes v Importações de dados fortes v Exportações de procedimentos fortes v Exportações de procedimentos fracos v Importações de procedimentos

Um objeto de módulo ILE pode exportar procedimentos ou itens de dados para outros módulos. E um objeto de módulo ILE pode importar procedimentos (de referência) ou itens de dados de outros módulos. Ao usar um objeto de módulo no comando CRTSRVPGM para criar um programa de serviço, suas exportações opcionalmente exportam do programa de serviço. (Consulte "Exportar parâmetro no comando CRTSRVPGM" na página 65.) A força (forte ou fraca) de uma exportação depende da linguagem de programação. A força determina quando o suficiente é conhecido sobre uma exportação para definir suas características, como o tamanho de um item de dados. As características de uma exportação forte são definidas no momento da ligação. A força das exportações afeta a resolução do símbolo.

v O aglutinante utiliza as características da exportação forte, se uma ou mais exportações fracas tiverem o mesmo nome.

v Se uma exportação fraca não tiver o mesmo nome que uma exportação forte, você não poderá definir suas características até o momento da ativação. No momento da ativação, se existirem várias exportações fracas com o mesmo nome, o programa usará a maior. Isso é verdade, a menos que uma exportação fraca já ativada com o mesmo nome já tenha definido suas características.

v No momento da vinculação, se um diretório de vinculação for usado e as exportações fracas corresponderem a importações fracas, elas serão vinculadas. No entanto, o diretório de associação é pesquisado somente enquanto houver importações não resolvidas a serem resolvidas. Depois que todas as importações forem resolvidas, a pesquisa pelas entradas do diretório de vinculação será interrompida. Exportações fracas duplicadas não são sinalizadas como variáveis ou procedimentos duplicados. A ordem dos itens no diretório de vinculação é muito importante.

Você pode exportar exportações fracas para fora de um objeto de programa ou programa de serviço para resolução no momento da ativação. Isso se opõe a exportações fortes que você exporta somente fora de um programa de serviço e somente no momento da associação.

No entanto, não é possível exportar exportações fortes para fora de um objeto de programa. Você pode exportar exportações de procedimento forte para fora de um programa de serviço para satisfazer uma das seguintes condições no momento da ligação:

v Importa em um programa que vincula o programa de serviço por referência. v Importações em outros programas de serviço que estão vinculados por referência a esse programa.

Os programas de serviço definem sua interface pública por meio da vinculação do idioma de origem.

Você pode tornar as exportações de procedimento fraco parte da interface pública de um programa de serviço por meio do idioma de origem de associação. No entanto, exportar uma exportação de procedimento fraco do programa de serviço por meio do idioma de origem de vinculação não o marca mais como fraco. É tratado como um forte procedimento de exportação.

Você só pode exportar dados fracos para um grupo de ativação. Você não pode torná-lo parte da interface pública que é exportada do programa de serviço por meio do uso da linguagem de origem do fichário. A especificação de dados fracos no idioma de origem do fichário faz com que a associação falhe.

A Tabela 8 resume os tipos de importações e exportações que são suportados por alguns dos idiomas ILE:

*Tabela 8. Importações e exportações suportadas por idiomas ILE*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ILE  Idiomas | Exportações de dados fracas | Importações de dados fracas | Exportações de dados fortes | Importações de dados fortes | Forte  Procedimento  Exportações | Fraco  Procedimento  Exportações | Procedimento Importações |
| RPG IV | Não | Não | Sim | Sim | Sim | Não | Sim |
| COBOL2 | Sim 3 | Sim 3 | Não | Não | Sim1 | Não | Sim |
| CL | Não | Não | Não | Não | Sim 1 | Não | Sim |
| C | Não | Não | Sim | Sim | Sim | Não | Sim |
| C++ | Não | Não | Sim | Sim | Sim | Sim | Sim |
| **Nota:**   1. COBOL e CL permitem que apenas um procedimento seja exportado do módulo. 2. O COBOL usa o modelo de dados fraco. Os itens de dados que são declarados como externos tornam-se exportações fracas e importações fracas para esse módulo. 3. COBOL requer a opção NOMONOPRC. Sem essa opção, as letras minúsculas são automaticamente convertidas em maiúsculas. | | | | | | | |

Para obter informações sobre quais declarações se tornam importações e exportações para um determinado idioma, consulte um dos seguintes livros:

v

ILE RPG Pr

Guia do ogrammer



v

ILE COBOL |

Pr

Guia do ogrammer



v

ILE C/C++ P

[r](http://publib.boulder.ibm.com/infocenter/iseries/v6r1m0/topic/rzaha/sc092712.pdf)

Guid do ogrammer

[e](http://publib.boulder.ibm.com/infocenter/iseries/v6r1m0/topic/rzaha/sc092712.pdf)



**Linguagem do fichário**

A **linguagem do fichário** é um pequeno conjunto de comandos não executáveis que define as exportações de um programa de serviço. A linguagem do fichário permite que o verificador de sintaxe do utilitário de entrada de origem (SEU) solicite e valide a entrada quando um tipo de origem BND é especificado.

**Nota:** Não é possível usar o tipo de verificação de sintaxe SEU BND para um arquivo de origem do fichário que contém curinga. Você também não pode usá-lo para um arquivo de origem do fichário que contém nomes com mais de 254 caracteres.

O idioma do fichário consiste em uma lista dos seguintes comandos:

1. Comando Iniciar Exportação de Programa (STRPGMEXP), que identifica o início de uma lista de exportações de um programa de serviço
2. Comandos Exportar Símbolo (EXPORT), cada um dos quais identifica um nome de símbolo disponível para ser exportado de um programa de serviço
3. Comando End Program Export (ENDPGMEXP), que identifica o final de uma lista de exportações de um programa de serviço

A Figura 32 é uma amostra da linguagem do fichário em um arquivo de origem:

STRPGMEXP PGMLVL(\*CURRENT) LVLCHK(\*SIM) .

.

SÍMBOLO DE EXPORTAÇÃO (p1)

SÍMBOLO DE EXPORTAÇÃO ('P2')

SÍMBOLO DE EXPORTAÇÃO («P3») .

.

ENDPGMEXP .

.

.

STRPGMEXP PGMLVL(\*PRV) .

.

SÍMBOLO DE EXPORTAÇÃO(P1) SÍMBOLO DE EXPORTAÇÃO(«P2») .

.

ENDPGMEXP

*Figura 32. Exemplo de linguagem de fichário em um arquivo de origem*

Usando o comando Recover Binder Source (RTVBNDSRC), você pode gerar a origem do idioma do fichário com base nas exportações de um ou mais módulos ou programas de serviço.

**Assinatura**

Os símbolos identificados entre um par STRPGMEXP PGMLVL(\*CURRENT) e ENDPGMEXP definem a interface pública para um programa de serviço. Essa interface pública é representada por uma *assinatura*. Uma assinatura é um valor que identifica a interface suportada por um programa de serviço.

**Observação:** Não confunda as assinaturas discutidas neste tópico com assinaturas *de objeto digital*. As assinaturas digitais em objetos IBM i garantem a integridade do software e dos dados. Eles também atuam como um impedimento para a adulteração de dados ou a modificação não autorizada de um objeto. A assinatura também fornece identificação positiva da origem dos dados. Para obter mais informações sobre assinaturas de objetos digitais, consulte a categoria Segurança de informações no Centro de Informações do IBM i.

Se você optar por especificar uma assinatura explícita, sua fonte de idioma do fichário só precisará ter um bloco de exportação; novas exportações podem ser adicionadas ao final da lista de exportações. Se você optar por não especificar uma assinatura explícita, o fichário gerará uma assinatura da lista de nomes de itens de procedimento e dados a serem exportados e da ordem em que eles são especificados. Você deve adicionar um novo bloco de exportação à origem do fichário sempre que adicionar novas exportações ao programa de serviço.

**Nota:** Para evitar alterações incompatíveis em um programa de serviço, os nomes de itens de dados e procedimentos existentes não devem ser removidos ou reorganizados na origem do idioma do fichário. Os blocos de exportação adicionais devem conter os mesmos símbolos na mesma ordem que os blocos de exportação existentes. Símbolos adicionais devem ser adicionados apenas ao final da lista. Essa regra se aplica se você especificar uma assinatura explícita ou se permitir que o fichário gere uma nova assinatura.

Não é possível remover uma exportação de programa de serviço de uma forma compatível com programas existentes e programas de serviço, pois essa exportação pode ser necessária para programas ou programas de serviço vinculados a esse programa de serviço.

Se uma alteração incompatível for feita em um programa de serviço, os programas existentes que permanecem vinculados a ele podem não funcionar mais corretamente. Uma alteração incompatível em um programa de serviço só pode ser feita se puder ser garantido que todos os programas e programas de serviço vinculados a ele sejam recriados com o comando CRTPGM ou CRTSRVPGM após a alteração incompatível ser feita.

Uma assinatura não valida a interface para um procedimento específico dentro de um programa de serviço. Uma alteração incompatível na interface de um determinado procedimento requer que todos os módulos que chamam o procedimento sejam recompilados e todos os programas e programas de serviço que contêm esses módulos sejam recriados com CRTPGM ou CRTSRVPGM.

**Iniciar Exportação de Programa e Encerrar Comandos de Exportação de Programa**

O comando Iniciar exportação de programa (STRPGMEXP) identifica o início de uma lista de exportações de um programa de serviço. O comando End Program Export (ENDPGMEXP) identifica o final de uma lista de exportações de um programa de serviço.

Vários pares STRPGMEXP e ENDPGMEXP especificados em um arquivo de origem fazem com que várias assinaturas sejam criadas. A ordem em que os pares STRPGMEXP e ENDPGMEXP ocorrem não é significativa.

#### Parâmetro de nível de programa no comando STRPGMEXP

Apenas um comando STRPGMEXP pode especificar PGMLVL(\*CURRENT), mas não precisa ser o primeiro

Comando STRPGMEXP. Todos os outros comandos STRPGMEXP dentro de um arquivo de origem devem especificar PGMLVL(\*PRV). A assinatura atual representa qualquer comando STRPGMEXP que tenha PGMLVL(\*CURRENT) especificado.

#### Parâmetro de assinatura no comando STRPGMEXP

O parâmetro signature (SIGNATURE) permite que você especifique explicitamente uma assinatura para um programa de serviço. A assinatura explícita pode ser uma cadeia de caracteres hexadecimal ou uma cadeia de caracteres. Talvez você queira considerar a especificação explícita de uma assinatura por um dos seguintes motivos:

v O fichário pode gerar uma assinatura compatível que você não deseja. Uma assinatura é baseada nos nomes das exportações especificadas e em seu pedido. Portanto, se dois blocos de exportação tiverem as mesmas exportações na mesma ordem, eles terão a mesma assinatura. Como o provedor de programa de serviço, você pode saber que as duas interfaces não são compatíveis (porque, por exemplo, suas listas de parâmetros são diferentes). Nesse caso, você pode especificar explicitamente uma nova assinatura em vez de fazer com que o fichário gere a assinatura compatível. Se você fizer isso, você cria uma incompatibilidade em seu programa de serviço, forçando alguns ou todos os clientes a recompilar.

v O fichário pode gerar uma assinatura incompatível que você não deseja. Se dois blocos de exportação tiverem exportações diferentes ou uma ordem diferente, eles terão assinaturas diferentes. Se, como provedor de programa de serviço, você souber que as duas interfaces são realmente compatíveis (porque, por exemplo, um nome de função foi alterado, mas ainda é a mesma função), você pode especificar explicitamente a mesma assinatura gerada anteriormente pelo fichário em vez de fazer com que o fichário gere uma assinatura incompatível. Se você especificar a mesma assinatura, manterá uma compatibilidade no programa de serviço, permitindo que os clientes usem o programa de serviço sem revinculação.

O valor padrão para o parâmetro signature, \*GEN, faz com que o fichário gere uma assinatura a partir de símbolos exportados.

Você pode determinar os valores de assinatura para um programa de serviço usando o comando Display Service Program (DSPSRVPGM) e especificando DETAIL(\*SIGNATURE).

#### Parâmetro Level Check no comando STRPGMEXP

O parâmetro level check (LVLCHK) no comando STRPGMEXP especifica se o fichário verificará automaticamente a interface pública para um programa de serviço. Especificar LVLCHK(\*YES) ou usar o valor padrão LVLCHK(\*YES) faz com que o fichário examine a assinatura em tempo de execução. O sistema verifica se o valor corresponde ao valor conhecido pelos clientes do programa de serviço. Se os valores corresponderem, os clientes do programa de serviço poderão usar a interface pública sem revincular ao programa de serviço.

Use o valor LVLCHK(\*NO) com cautela. Se você não puder controlar a interface pública, poderão ocorrer erros de tempo de execução ou de ativação. Consulte "Binder Language Errors" na página 166 para obter uma explicação dos erros comuns que podem ocorrer ao usar o idioma do fichário.

**Comando Exportar Símbolo**

O comando Exportar Símbolo (EXPORT) identifica um nome de símbolo disponível para ser exportado de um programa de serviço.

Se os símbolos exportados contiverem letras minúsculas, o nome do símbolo deve ser incluído em apóstrofos, como na Figura 32 na página 68. Se os apóstrofos não forem usados, o nome do símbolo será convertido em todas as letras maiúsculas. No exemplo, o fichário procura uma exportação chamada P1, não p1.

Os nomes dos símbolos também podem ser exportados através do uso de caracteres curinga (<<< ou >>>). Se um nome de símbolo existir e corresponder ao curinga especificado, o nome do símbolo será exportado. Se qualquer uma das seguintes condições existir, um erro será sinalizado e o programa de serviço não será criado:

v Nenhum nome de símbolo corresponde ao curinga especificado v Mais de um nome de símbolo corresponde ao curinga especificado

v Um nome de símbolo corresponde ao curinga especificado, mas não está disponível para exportação

As subcadeias de caracteres na especificação curinga devem estar entre aspas.

As assinaturas são determinadas pelos caracteres nas especificações curinga. Alterar a especificação curinga altera a assinatura mesmo que a especificação curinga alterada corresponda à mesma exportação. Por exemplo, as duas especificações curinga "r">>> e "ra">>> exportam o símbolo "taxa", mas criam duas assinaturas diferentes. Portanto, é altamente recomendável que você use uma especificação curinga que seja o mais semelhante possível ao símbolo de exportação.

**Nota:** Não é possível usar o tipo de verificação de sintaxe SEU BND para um arquivo de origem do fichário que contém curinga.

#### Exemplos de símbolos de exportação curinga

Para os exemplos a seguir, suponha que a lista de símbolos de exportações possíveis consista em:

interest\_rate prime\_rate internacionais

Os exemplos a seguir mostram qual exportação é escolhida ou por que ocorre um erro:

**SÍMBOLO DE EXPORTAÇÃO ("interesse">>>)**

Exporta o símbolo "interest\_rate" porque é o único símbolo que começa com "juros".

##### SÍMBOLO DE EXPORTAÇÃO ("I">>>"TAXA">>>)

Exporta o símbolo "interest\_rate" porque é o único símbolo que começa com "i" e, posteriormente, contém "taxa".

##### SÍMBOLO DE EXPORTAÇÃO (<<<"i">>>"taxa")

Resulta em um erro "Várias correspondências para especificação curinga". Tanto "prime\_rate" quanto "interest\_rate" contêm um "i" e, posteriormente, terminam em "taxa".

##### SÍMBOLO DE EXPORTAÇÃO ("INTER">>>"PRIME")

Resulta em um erro "Não corresponde à especificação curinga". Nenhum símbolo começa com "inter" e, posteriormente, termina em "prime".

##### SÍMBOLO DE EXPORTAÇÃO (<<<)

Resulta em um erro "Várias correspondências para especificação curinga". Este símbolo corresponde a todos os três símbolos e, portanto, não é válido. Uma instrução de exportação pode resultar em apenas um símbolo exportado.

**Exemplos de linguagem Binder**

Como exemplo de uso da linguagem do fichário, suponha que você esteja desenvolvendo um aplicativo financeiro simples com os seguintes procedimentos:

v Procedimento tarifário

Calcula uma Interest\_Rate, dados os valores de Loan\_Amount, Term\_of\_Payment e Payment\_Amount. v Procedimento do montante

Calcula o Loan\_Amount, dados os valores de Interest\_Rate, Term\_of\_Payment e Payment\_Amount. v Procedimento de pagamento

Calcula o Payment\_Amount, dados os valores de Interest\_Rate, Term\_of\_Payment e Loan\_Amount. v Procedimento a termo

Calcula o Term\_of\_Payment, dados os valores de Interest\_Rate, Loan\_Amount e Payment\_Amount.

Algumas das listagens de saída para este aplicativo são mostradas no Apêndice A, "Listagem de saída do comando CRTPGM, CRTSRVPGM, UPDPGM ou UPDSRVPGM", na página 157.

Nos exemplos de linguagem do fichário, cada módulo contém mais de um procedimento. Os exemplos se aplicam até mesmo a módulos que contêm apenas um procedimento.

#### Exemplo 1 da linguagem do fichário

O idioma do fichário para os procedimentos Taxa, Valor, Pagamento e Prazo tem a seguinte aparência:

ARQUIVO: MYLIB/QSRVSRC MEMBRO: FINANCEIRO

STRPGMEXP PGMLVL(\*CURRENT) EXPORT SYMBOL('Termo')

SÍMBOLO DE EXPORTAÇÃO ('Taxa')

SÍMBOLO DE EXPORTAÇÃO ('Quantidade')

SÍMBOLO DE EXPORTAÇÃO ('Pagamento')

ENDPGMEXP

Algumas decisões iniciais de projeto foram tomadas, e três módulos (MONEY, RATES e CALCS) fornecem os procedimentos necessários.

Para criar o programa de serviço representado na Figura 33 na página 72, o idioma do fichário é especificado no seguinte comando CRTSRVPGM:

CRTSRVPGM SRVPGM (MYLIB/FINANCEIRO)

MÓDULO (MYLIB/MONEY MYLIB/TAXAS MYLIB/CALCS)

EXPORTAR(\*SRCFILE)

SRCFILE(MYLIB/QSRVSRC)

SRCMBR(\*SRVPGM)

Observe que o arquivo de origem QSRVSRC na biblioteca MYLIB, especificado no parâmetro SRCFILE, é o arquivo que contém a origem do idioma do fichário.

Observe também que nenhum diretório de associação é necessário porque todos os módulos necessários para criar o programa de serviço são especificados no parâmetro MODULE.

**Programa de Serviço**

**MYLIB/FINANCEIRO**

Módulo DINHEIRO

Assinatura atual = Sig 123

TAXAS DO MÓDULO

Módulo CALCS

Valor do Procedimento

Procedimento de Pagamento

Termo do Procedimento

Taxa de Procedimento

Procedimento CALC1

Procedimento CALC2

Prazo

Táxons

Quantidade

Pagamento

**Público**

**Interface**

RV2W1051-3

*Figura 33. Criando um programa de serviço usando a linguagem do fichário*

#### Exemplo 2 de linguagem de fichário

À medida que o progresso é feito no desenvolvimento do aplicativo, um programa chamado BANKER é escrito. BANKER precisa usar o procedimento chamado Pagamento no programa de serviço chamado FINANCEIRO. A aplicação resultante com o programa BANKER é mostrada na Figura 34 na página 73.

**Programa de Serviço**

**MYLIB/FINANCEIRO**

**Programa BANQUEIRO**

Módulo DINHEIRO

Assinatura atual = Sig 123

TAXAS DO MÓDULO

Módulo M1

Módulo CALCS

Valor do Procedimento

Procedimento de Pagamento

Termo do Procedimento

Taxa de Procedimento

Procedimento CALC1

Procedimento CALC2

Prazo

Táxons

Quantidade

Pagamento

CallPrc Pagamento

MYLIB/FINANCEIRO

Pagamento = 4º slot

Assinatura = Sig 123

**Público**

**Interface**

RV2W1053-4

*Figura 34. Usando o Programa de Serviço FINANCEIRO*

Quando o programa BANKER foi criado, o programa de serviços MYLIB/FINANCIAL foi fornecido no parâmetro BNDSRVPGM. Verificou-se que o símbolo Pagamento foi exportado do quarto slot da interface pública do programa de serviços FINANCEIROS. A assinatura atual do MYLIB/FINANCIAL, juntamente com o slot associado à interface de pagamento, é salva com o programa BANKER.

Durante o processo de preparar o BANKER para ser executado, a ativação verifica o seguinte: v O programa de serviço FINANCEIRO na biblioteca MYLIB pode ser encontrado. v O programa de serviço ainda suporta a assinatura (SIG 123) salva no BANKER.

Essa verificação de assinatura verifica se a interface pública usada pelo BANKER quando ele foi criado ainda é válida em tempo de execução.

Como mostrado na Figura 34, no momento em que BANKER é chamado, MYLIB/FINANCIAL ainda suporta a interface pública usada por BANKER. Se a ativação não puder encontrar uma assinatura correspondente no MYLIB/FINANCIAL ou no programa de serviço MYLIB/FINANCIAL, ocorrerá o seguinte:

BANKER não consegue ser ativado. Uma mensagem de erro é emitida.

#### Exemplo 3 da linguagem do fichário

À medida que a candidatura continua a crescer, são necessários dois novos procedimentos para completar o nosso pacote financeiro. Os dois novos procedimentos, OpenAccount e CloseAccount, abrem e fecham as contas, respectivamente. As seguintes etapas precisam ser executadas para atualizar o MYLIB/FINANCIAL de modo que o programa BANKER não precise ser recriado:

1. Escreva os procedimentos OpenAccount e CloseAccount.
2. Atualize o idioma do fichário para especificar os novos procedimentos.

A linguagem do fichário atualizada oferece suporte aos novos procedimentos. Também permite que os programas ILE existentes ou os programas de serviço que usam o programa de serviços FINANCEIROS permaneçam inalterados. A linguagem do fichário tem esta aparência:

ARQUIVO: MYLIB/QSRVSRC MEMBRO: FINANCEIRO

STRPGMEXP PGMLVL(\*CURRENT) EXPORT SYMBOL('Termo')

SÍMBOLO DE EXPORTAÇÃO ('Taxa')

SÍMBOLO DE EXPORTAÇÃO ('Quantidade')

SÍMBOLO DE EXPORTAÇÃO ('Pagamento')

SÍMBOLO DE EXPORTAÇÃO ('OpenAccount')

SÍMBOLO DE EXPORTAÇÃO ('CloseAccount')

ENDPGMEXP

STRPGMEXP PGMLVL(\*PRV) SÍMBOLO DE EXPORTAÇÃO ('Termo')

SÍMBOLO DE EXPORTAÇÃO ('Taxa')

SÍMBOLO DE EXPORTAÇÃO ('Quantidade')

SÍMBOLO DE EXPORTAÇÃO ('Pagamento')

ENDPGMEXP

Quando uma operação de atualização para um programa de serviço é necessária para fazer o seguinte: v Oferecer suporte a novos procedimentos ou itens de dados v Permitir que os programas existentes e os programas de serviço que usam o programa de serviço alterado permaneçam inalterados, uma das duas alternativas deve ser escolhida. A primeira alternativa é executar as seguintes etapas:

1. Duplique o bloco STRPGMEXP, ENDPGMEXP que contém PGMLVL(\*CURRENT).
2. Altere o valor PGMLVL(\*CURRENT) duplicado para PGMLVL(\*PRV).
3. No comando STRPGMEXP que contém PGMLVL(\*CURRENT), adicione ao final da lista os novos procedimentos ou itens de dados a serem exportados.
4. Salve as alterações no arquivo de origem.
5. Crie ou recrie os módulos novos ou alterados.
6. Crie o programa de serviço a partir dos módulos novos ou alterados usando a linguagem de fichário atualizada.

A segunda alternativa é aproveitar o parâmetro signature no comando STRPGMEXP e adicionar novos símbolos no final do bloco de exportação:

STRPGMEXP PGMVAL(\*CURRENT) SIGNATURE('123') EXPORT SYMBOL('Term') .

.

.

SÍMBOLO DE EXPORTAÇÃO ('OpenAccount')

SÍMBOLO DE EXPORTAÇÃO ('CloseAccount')

ENDPGMEXP

Para criar o programa de serviço avançado mostrado na Figura 35 na página 75, o idioma do fichário atualizado mostrado em "Binder Language Example 3" é usado no seguinte comando CRTSRVPGM:

CRTSRVPGM SRVPGM (MYLIB/FINANCEIRO)

MÓDULO(MYLIB/MONEY MYLIB/RATES MYLIB/CALCS MYLIB/ACCOUNTS)) EXPORT(\*SRCFILE)

SRCFILE(MYLIB/QSRVSRC)

SRCMBR(\*SRVPGM)

**Programa de Serviço**

**MYLIB/FINANCEIRO**

**Programa BANQUEIRO**

Módulo Dinheiro

Módulo M1

Assinatura Anterior = Sig 123

Assinatura atual = Sig 456

TAXAS DO MÓDULO

Módulo |

CONTAS

Módulo CALCS

Procedimento

Quantidade

Procedimento

Pagamento

Procedimento

Táxons

Procedimento

Prazo

Procedimento

Procedimento

OpenAccount

FecharConta |

Procedimento

CALC2

Procedimento

CALC1

Prazo

Táxons

Quantidade

Pagamento

OpenAccount

FecharConta |

CallPrc Pagamento

MYLIB/FINANCEIRO

Pagamento = 4º slot

Assinatura = Sig 123

**Público**

**Interface**

RV2W1052-4

*Figura 35. Atualizando um programa de serviço usando o idioma do fichário*

O programa BANKER não precisa ser alterado porque a assinatura anterior ainda é suportada. (Veja a assinatura anterior no programa de serviço MYLIB/FINANCIAL e a assinatura salva no BANKER.) Se BANKER fosse recriado pelo comando CRTPGM, a assinatura que é salva com BANKER seria a assinatura atual do programa de serviço FINANCIAL. A única razão para recriar o programa BANKER é se o programa usou um dos novos procedimentos fornecidos pelo programa de serviço FINANCIAL. A linguagem do fichário permite que você aprimore o programa de serviço sem alterar os programas ou programas de serviço que usam o programa de serviço alterado.

#### Exemplo 4 da linguagem do fichário

Depois de enviar o programa de serviços FINANCEIROS atualizado, você recebe uma solicitação para criar uma taxa de juros com base no seguinte:

Os parâmetros atuais do procedimento Taxa

O histórico de crédito do candidato

Um quinto parâmetro, chamado Credit\_History, deve ser adicionado na chamada para o procedimento Rate. Credit\_History atualiza o parâmetro Interest\_Rate que é retornado do procedimento Rate. Outro requisito é que os programas ILE existentes ou os programas de serviço que usam o programa de serviço FINANCEIRO não devem ter que ser alterados. Se o idioma não oferecer suporte à passagem de um número variável de parâmetros, parece difícil fazer o seguinte:

v Atualizar o programa de serviço

v Evite recriar todos os outros objetos que usam o programa de serviço FINANCEIRO

Felizmente, no entanto, há uma maneira de fazer isso. A linguagem de fichário a seguir oferece suporte ao procedimento Rate atualizado. Ele ainda permite que os programas ILE existentes ou os programas de serviço que usam o programa de serviços FINANCEIROS permaneçam inalterados.

ARQUIVO: MYLIB/QSRVSRC MEMBRO: FINANCEIRO

|  |  |
| --- | --- |
| STRPGMEXP PGMLVL(\*ATUAL)  SÍMBOLO DE EXPORTAÇÃO ('Termo')  SÍMBOLO DE EXPORTAÇÃO('Old\_Rate') /\* Procedimento de taxa original com quatro parâmetros \*/ | |
| SÍMBOLO DE EXPORTAÇÃO ('Quantidade')  SÍMBOLO DE EXPORTAÇÃO ('Pagamento')  SÍMBOLO DE EXPORTAÇÃO ('OpenAccount')  SÍMBOLO DE EXPORTAÇÃO ('CloseAccount') |  |
| SÍMBOLO DE EXPORTAÇÃO ('Taxa')  ENDPGMEXP  STRPGMEXP PGMLVL(\*PRV) SÍMBOLO DE EXPORTAÇÃO ('Termo')  SÍMBOLO DE EXPORTAÇÃO ('Taxa')  SÍMBOLO DE EXPORTAÇÃO ('Quantidade')  SÍMBOLO DE EXPORTAÇÃO ('Pagamento')  SÍMBOLO DE EXPORTAÇÃO ('OpenAccount')  SÍMBOLO DE EXPORTAÇÃO('CloseAccount') ENDPGMEXP | /\* Novo procedimento de taxa que suporta + um quinto parâmetro, Credit\_History \*/ |

STRPGMEXP PGMLVL(\*PRV) SÍMBOLO DE EXPORTAÇÃO ('Termo')

SÍMBOLO DE EXPORTAÇÃO ('Taxa')

SÍMBOLO DE EXPORTAÇÃO ('Quantidade')

SÍMBOLO DE EXPORTAÇÃO ('Pagamento')

O símbolo original Rate foi renomeado Old\_Rate mas permanece na mesma posição relativa dos símbolos a serem exportados. Isso é importante lembrar.

Um comentário é associado ao símbolo Old\_Rate. Um comentário é tudo entre /\* e \*/. O fichário ignora comentários na origem do idioma do fichário ao criar um programa de serviço.

O novo procedimento Rate, que suporta o parâmetro adicional de Credit\_History, também deve ser exportado. Este procedimento atualizado é adicionado ao final da lista de exportações.

As duas maneiras a seguir podem lidar com o procedimento de taxa original:

v Renomeie o procedimento Rate original que oferece suporte a quatro parâmetros como Old\_Rate. Duplicar o

Old\_Rate procedimento (chamando-o de Taxa). Atualize o código para oferecer suporte ao quinto parâmetro do Credit\_History.

v Atualize o procedimento Rate original para dar suporte ao quinto parâmetro de Credit\_History. Crie um novo procedimento chamado Old\_Rate. Old\_Rate suporta os quatro parâmetros originais de Rate. Ele também chama o novo procedimento Rate atualizado com um quinto parâmetro fictício.

Esse é o método preferido porque a manutenção é mais simples e o tamanho do objeto é menor.

Usando a linguagem de fichário atualizada e um novo módulo RATES que oferece suporte aos procedimentos Taxa, Prazo e Old\_Rate, você cria o seguinte programa de serviço FINANCEIRO:

**Programa de Serviços MYLIB/FINANCIAL**

Módulo DINHEIRO

Assinatura atual = Sig 789

assinaturas anteriores = sig 456, sig 123

TAXAS DO MÓDULO

Módulo |

CONTAS

Módulo CALCS

Procedimento

Procedimento

Quantidade

Pagamento

Procedimento

Procedimento

Procedimento

Prazo

Old\_Rate

Táxons

Procedimento

Procedimento

OpenAccount

FecharConta |

CALC1

CALC2

Procedimento

Procedimento

Prazo

Old\_Rate

Quantidade

Pagamento

OpenAccount

FecharConta |

Táxons

**Público**

**Interface**

RV2W1055-2

*Figura 36. Atualizando um programa de serviço usando o idioma do fichário*

Os programas ILE e os programas de serviço que usam o procedimento Rate original do programa de serviços FINANCEIROS vão para o slot 2. Isso direciona a chamada para o procedimento Old\_Rate, o que é vantajoso porque Old\_Rate manipula os quatro parâmetros originais. Se qualquer um dos programas ILE ou programas de serviço que usaram o procedimento Rate original precisar ser recriado, siga um destes procedimentos:

v Para continuar a usar o procedimento Rate original de quatro parâmetros, chame o procedimento Old\_Rate em vez do procedimento Rate.

v Para usar o novo procedimento Rate, adicione o quinto parâmetro, Credit\_History, a cada chamada para o procedimento Rate.

Quando uma atualização para um programa de serviço deve atender aos seguintes requisitos: v Oferecer suporte a um procedimento que alterou o número de parâmetros que ele pode processar v Permitir que programas existentes e programas de serviço que usam o programa de serviço alterado permaneçam inalterados

as seguintes etapas precisam ser executadas:

1. Duplique o bloco STRPGMEXP, ENDPGMEXP que contém PGMLVL(\*CURRENT).
2. Altere o valor PGMLVL(\*CURRENT) duplicado para PGMLVL(\*PRV).
3. No comando STRPGMEXP que contém PGMLVL(\*CURRENT), renomeie o nome do procedimento original, mas deixe-o na mesma posição relativa.

Neste exemplo, Rate foi alterado para Old\_Rate mas deixado na mesma posição relativa na lista de símbolos a serem exportados.

1. No comando STRPGMEXP que tem PGMLVL(\*CURRENT), coloque o nome do procedimento original no final da lista que oferece suporte a um número diferente de parâmetros.

Neste exemplo, Rate é adicionado ao final da lista de símbolos exportados, mas esse procedimento Rate oferece suporte ao parâmetro adicional Credit\_History.

1. Salve as alterações no arquivo de origem do idioma do fichário.
2. No arquivo que contém o código-fonte, aprimore o procedimento original para oferecer suporte ao novo parâmetro.

No exemplo, isso significa alterar o procedimento Rate existente para dar suporte ao quinto parâmetro de Credit\_History.

1. Um novo procedimento é criado que manipula os parâmetros originais como entrada e chama o novo procedimento com um parâmetro extra fictício.

No exemplo, isso significa adicionar o procedimento Old\_Rate que manipula os parâmetros originais e chamar o novo procedimento Rate com um quinto parâmetro fictício.

1. Salve as alterações de código-fonte do idioma do fichário.
2. Crie os objetos de módulo com os procedimentos novos e alterados.
3. Crie o programa de serviço a partir dos módulos novos e alterados usando a linguagem de fichário atualizada.

**Alterando programas**

| Os comandos Programa de Alteração ( CHGPGM) e Programa de Serviço de Alteração (CHGSRVPGM) alteram o | atributos de um programa e programa de serviço sem a necessidade de recompilação. Algumas das | mutáveis os atributos seguem:

| v O atributo de otimização.

| v O atributo de perfil de usuário.

| v Use o atributo de autoridade adotado.

| v O atributo de dados de criação de perfil.

| v O texto do programa.

| v Opções de Código Interno Licenciado.

| v Modelo de armazenamento (somente de \*SNGLVL para \*INHERIT).

O usuário também pode forçar a recriação de um programa, mesmo que os atributos especificados sejam os mesmos que os atributos atuais. Faça isso especificando o parâmetro de recreação de programa de força (FRCCRT) com um valor de \*YES.

O parâmetro de recreação do programa de força (FRCCRT) também pode ser especificado com os valores de \*NO e \*NOCRT. Esses valores determinam se os atributos de programa solicitados são realmente alterados quando a alteração requer que o programa seja recriado. Modificar os seguintes atributos de programa pode fazer com que o programa seja recriado:

v O prompt do programa Optimize (parâmetro OPTIMIZ) v O prompt Usar autoridade adotada (parâmetro USEADPAUT) v O prompt de dados de criação de perfil (parâmetro PRFDTA) v O prompt de perfil do usuário (parâmetro USRPRF) v Prompt de opções de código interno licenciado (parâmetro LICOPT) | v Prompt do modelo de armazenamento (parâmetro STGMDL)

Um valor de \*NO para o parâmetro de recriação de programa de força (FRCCRT) significa que a recriação não é forçada, mas se um dos atributos do programa que exigem recriação foi alterado, o programa é recriado. Essa opção permite que o sistema determine se uma alteração é necessária.

Recriar um programa com CHGPGM ou CHGSRVPGM enquanto um ou mais trabalhos estão usando o programa faz com que uma exceção "Objeto destruído" ocorra e esses trabalhos podem falhar. Alterando o padrão de comando para o parâmetro de recriação de programa de força (FRCCRT) para \*NOCRT, você pode impedir que isso aconteça inadvertidamente.

Você pode usar o parâmetro number of threads (NBRTHD) para aproveitar os ciclos de unidade de processamento disponíveis, especialmente em um sistema multiprocessador.

**Atualizações do programa**

Depois que um objeto de programa ILE ou programa de serviço é criado, talvez seja necessário corrigir um erro nele ou adicionar um aprimoramento a ele. No entanto, depois de fazer a manutenção do objeto, ele pode ser tão grande que o envio de todo o objeto para seus clientes é difícil ou caro.

Você pode reduzir o tamanho da remessa usando o comando Programa de Atualização (UPDPGM) ou Programa de Serviço de Atualização (UPDSRVPGM). Esses comandos substituem apenas os módulos especificados e somente os módulos alterados ou adicionados precisam ser enviados aos seus clientes.

Se você usar o processo PTF, um programa de saída contendo uma ou mais chamadas para o UPDPGM ou

Os comandos UPDSRVPGM podem ser usados para executar as funções de atualização. Vincular o mesmo módulo a vários objetos de programa ou programas de serviço requer a execução do comando UPDPGM ou UPDSRVPGM em cada objeto \*PGM e \*SRVPGM.

Por exemplo, consulte a Figura 37 na página 80.

**Programa de Serviços MYLIB/FINANCIAL**

Módulo DINHEIRO

Assinatura atual = Sig 789

assinaturas anteriores = sig 456, sig 123

TAXAS DO MÓDULO

Módulo |

CONTAS

Módulo CALCS

Procedimento

Procedimento

Quantidade

Pagamento

Procedimento

Procedimento

Procedimento

Prazo

Old\_Rate

Táxons

Procedimento

Procedimento

OpenAccount

FecharConta |

CALC1

CALC2

Procedimento

Procedimento

Prazo

Old\_Rate

Quantidade

Pagamento

OpenAccount

FecharConta |

Táxons

**Público**

**Interface**

RV3W105-0

Módulo DINHEIRO

Procedimento

Procedimento

Quantidade

Pagamento

*Figura 37. Substituindo um módulo em um programa de serviço*

Se um programa ou programa de serviço for atualizado enquanto permanecer ativado em outro trabalho, o trabalho continuará a usar a versão antiga do programa ou programa de serviço. Novas ativações usarão a versão atualizada do programa ou programa de serviço.

Os parâmetros allow update (ALWUPD) e allow \*SRVPGM library update (ALWLIBUPD) no comando CRTPGM ou CRTSRVPGM determinam se um objeto de programa ou programa de serviço pode ser atualizado. Especificando ALWUPD(\*NO), os módulos em um objeto de programa ou programa de serviço não podem ser substituídos pelo comando UPDPGM ou UPDSRVPGM. Especificando ALWUPD(\*YES) e

ALWLIBUPD(\*YES), você pode atualizar seu programa para usar um programa de serviço de uma biblioteca que não foi especificada anteriormente. Especificando ALWUPD(\*YES) e ALWLIBUPD(\*NO), você pode atualizar os módulos, mas não a biblioteca de programas de serviço acoplada. Você não pode especificar ALWUPD(\*NO) e ALWLIBUPD(\*YES) ao mesmo tempo.

**Parâmetros nos comandos UPDPGM e UPDSRVPGM**

Cada módulo especificado no parâmetro module substitui um módulo com o mesmo nome que está vinculado a um objeto de programa ou programa de serviço. Se mais de um módulo vinculado a um objeto de programa ou programa de serviço tiver o mesmo nome, o parâmetro de biblioteca de substituição (RPLLIB) será usado. Esse parâmetro especifica qual método é usado para selecionar o módulo a ser substituído. Se nenhum módulo com o mesmo nome já estiver vinculado a um objeto de programa ou programa de serviço, o objeto de programa ou programa de serviço não será atualizado.

O parâmetro de programa de serviço acoplado (BNDSRVPGM) especifica programas de serviço adicionais além daqueles aos quais o objeto de programa ou programa de serviço já está vinculado. Se um módulo de substituição contiver mais importações ou menos exportações do que o módulo que ele substitui, esses programas de serviço poderão ser necessários para resolver essas importações.

Com o parâmetro de biblioteca de programas de serviço (SRVPGMLIB), você pode especificar a biblioteca que armazena os programas de serviço acoplados. Cada vez que você executa os comandos UPDPGM ou UPDSRVPGM, os programas de serviço acoplados da biblioteca especificada são usados. O comando UPDPGM ou UPDSRVPGM permite que você altere a biblioteca se ALWLIBUPD(\*YES) for usado.

O parâmetro de diretório de vinculação (BNDDIR) especifica diretórios de vinculação que contêm módulos ou programas de serviço que também podem ser necessários para resolver importações extras.

O parâmetro de grupo de ativação (ACTGRP) especifica o nome do grupo de ativação a ser usado quando um programa ou programa de serviço é ativado. Esse parâmetro também permite que você altere o nome do grupo de ativação de um grupo de ativação nomeado.

**Módulo substituído por um módulo com menos importações**

Se um módulo for substituído por outro módulo com menos importações, o novo objeto de programa ou programa de serviço será sempre criado. No entanto, o objeto de programa atualizado ou programa de serviço contém um módulo isolado se as seguintes condições existirem:

v Devido às importações agora ausentes, um dos módulos vinculados a um objeto de programa ou programa de serviço não resolve mais nenhuma importação

v Esse módulo veio originalmente de um diretório de vinculação usado no CRTPGM ou CRTSRVPGM

comando

Programas com módulos isolados podem crescer significativamente ao longo do tempo. Para remover módulos que não resolvem mais nenhuma importação e que originalmente vieram de um diretório de associação, você pode especificar OPTION(\*TRIM) ao atualizar os objetos. No entanto, se você usar essa opção, as exportações que os módulos contêm não estão disponíveis para futuras atualizações do programa.

**Módulo substituído por um módulo com mais importações**

Se um módulo for substituído por um módulo com mais importações, o objeto de programa ou programa de serviço poderá ser atualizado se essas importações extras forem resolvidas, de acordo com o seguinte:

v O conjunto existente de módulos vinculados ao objeto. v Programas de serviço vinculados ao objeto.

v Diretórios de vinculação especificados no comando. Se um módulo em um desses diretórios de vinculação contiver uma exportação necessária, o módulo será adicionado ao programa ou programa de serviço. Se um programa de serviço em um desses diretórios de associação contiver uma exportação necessária, o programa de serviço será vinculado por referência ao programa ou programa de serviço.

v Diretórios de vinculação implícitos. Um diretório de associação **implícito** é um diretório de associação que contém exportações que podem ser necessárias para criar um programa que contém o módulo. Cada compilador ILE cria uma lista de diretórios de vinculação implícitos em cada módulo que cria.

Se essas importações extras não puderem ser resolvidas, a operação de atualização falhará, a menos que OPTION(\*UNRSLVREF) seja especificado no comando update.

**Módulo substituído por um módulo com menos exportações**

Se um módulo for substituído por outro módulo com menos exportações, a atualização ocorrerá se as seguintes condições existirem: v As exportações ausentes não são necessárias para a vinculação. v As exportações em falta não são exportadas para fora do programa de serviço no caso do UPDSRVPGM.

Se um programa de serviço for atualizado com EXPORT(\*ALL) especificado, uma nova lista de exportação será criada. A nova lista de exportação será diferente da lista de exportação original.

A atualização não ocorre se as seguintes condições existirem:

v Algumas importações não podem ser resolvidas devido à falta de exportações.

v Essas exportações ausentes não podem ser encontradas nos programas de serviço extras e nos diretórios de vinculação especificados no comando. v O idioma do fichário indica exportar um símbolo, mas a exportação está ausente.

**Módulo substituído por um módulo com mais exportações**

Se um módulo for substituído por outro módulo com mais exportações, a operação de atualização ocorrerá se todas as exportações extras tiverem o nome exclusivo. A exportação do programa de serviço será diferente se EXPORT(\*ALL) for especificado.

No entanto, se uma ou mais das exportações extras não forem nomeadas exclusivamente, os nomes duplicados podem causar um problema:

v Se OPTION(\*NODUPPROC) ou OPTION(\*NODUPVAR) for especificado no comando update, o objeto de programa ou programa de serviço não será atualizado.

v Se OPTION(\*DUPPROC) ou OPTION(\*DUPVAR) for especificado, a atualização ocorrerá, mas a exportação extra poderá ser usada em vez da exportação original de mesmo nome.

**Dicas para criar módulos, programas e programas de serviço**

Para criar e manter módulos, programas ILE e programas de serviço convenientemente, considere o seguinte:

v Siga uma convenção de nomenclatura para os módulos que serão copiados para criar um programa ou serviço

programa.

Uma estratégia de nomenclatura com um prefixo comum facilita a especificação genérica de módulos no parâmetro module.

v Para facilitar a manutenção, inclua cada módulo em apenas um programa ou programa de serviço. Se mais de um programa precisar usar um módulo, coloque-o em um programa de serviço. Dessa forma, se você tiver que redesenhar um módulo, você só precisa redesenhá-lo em um só lugar. v Para garantir sua assinatura, use a linguagem do fichário sempre que criar um programa de serviço.

A linguagem do fichário permite que o programa de serviço seja facilmente atualizado sem ter que recriar os programas de uso e os programas de serviço.

O comando Recover Binder Source (RTVBNDSRC) pode ser usado para ajudar a gerar a origem do idioma do fichário com base nas exportações de um ou mais módulos ou programas de serviço.

Se uma das seguintes condições existir:

* Um programa de serviço nunca mudará
* Os usuários do programa de serviço não se importam de alterar seus programas quando uma assinatura é alterada, você não precisa usar a linguagem do fichário. Como essa situação não é provável para a maioria dos aplicativos, considere o uso da linguagem do fichário para todos os programas de serviço.

v Se você receber uma mensagem CPF5D04 ao usar um comando de criação de programa como CRTPGM, CRTSRVPGM ou UPDPGM, mas seu programa ou programa de serviço ainda estiver criado, há duas explicações possíveis:

* 1. Seu programa é criado com OPTION(\*UNRSLVREF) e contém referências não resolvidas.
  2. Você está vinculado a um \*SRVPGM listado em \*BNDDIR QSYS/QUSAPIBD que é fornecido com \*PUBLIC \*EXCLUDE authority, e você não tem autoridade. Para ver quem está autorizado a um objeto, use o comando DSPOBJAUT. O sistema \*BNDDIR QUSAPIBD contém os nomes de \*SRVPGMs que fornecem APIs do sistema. Algumas dessas APIs são sensíveis à segurança, portanto, os \*SRVPGMs em que estão são fornecidos com a autoridade \*PUBLIC \*EXCLUDE. Esses \*SRVPGMs são agrupados no final do QUSAPIBD. Quando você estiver usando um programa de serviço \*PUBLIC \*EXCLUDE nesta lista, o fichário geralmente precisa examinar outros \*PUBLIC \*EXCLUDE \*SRVPGMs à frente do seu, e ele leva o CPF5D04.

Para evitar receber a mensagem CPF5D04, use um dos seguintes métodos:

* + Especifique explicitamente quaisquer \*SRVPGMs aos quais seu programa ou programa de serviço esteja vinculado. Para ver a lista de \*SRVPGMS aos quais seu programa ou serviço está vinculado, use DSPPGM ou DSPSRVPGM DETAIL(\*SRVPGM). Esses \*SRVPGMs podem ser especificados no parâmetro CRTPGM ou CRTSRVPGM BNDSRVPGM. Eles também podem ser colocados em um diretório de vinculação dado no CRTBNDRPG, CRTRPGMOD,

CRTBNDCBL, CRTPGM ou CRTSRVPGM BNDDIR parâmetro, ou de um RPG H-spec. Tomar essa ação garante que todas as referências sejam resolvidas antes que o \*PUBLIC \*EXCLUDE \*SRVPGMs em \*BNDDIR QUSAPIBD precise ser examinado.

* + Conceder \*autoridade PÚBLICA ou individual aos \*SRVPGMs listados nas mensagens do CPF5D04. Isso tem a desvantagem de autorizar os usuários a interfaces potencialmente sensíveis à segurança desnecessariamente.
  + Se OPTION(\*UNRSLVREF) for usado e seu programa contiver referências não resolvidas, verifique se todas as referências estão resolvidas.

v Se outras pessoas usarem um objeto de programa ou programa de serviço que você criar, especifique

OPTION(\*RSLVREF) ao criá-lo. Quando você estiver desenvolvendo um aplicativo, convém criar um objeto de programa ou programa de serviço com importações não resolvidas. No entanto, quando em produção, todas as importações devem ser resolvidas.

Se OPTION(\*WARN) for especificado, as referências não resolvidas serão listadas no log de trabalho que contém a solicitação CRTPGM ou CRTSRVPGM. Se você especificar uma listagem no parâmetro DETAIL, elas também serão incluídas na listagem do programa. Você deve manter o registro de trabalho ou a listagem.

v Ao projetar novos aplicativos, determine se os procedimentos comuns que devem entrar em um ou mais programas de serviço podem ser identificados.

É provavelmente mais fácil identificar e projetar procedimentos comuns para novas aplicações. Se você estiver convertendo um aplicativo existente para usar o ILE, pode ser mais difícil determinar procedimentos comuns para um programa de serviço. No entanto, tente identificar procedimentos comuns necessários para o aplicativo e tente criar programas de serviço contendo os procedimentos comuns.

v Ao converter um aplicativo existente em ILE, considere a criação de alguns programas grandes.

Com algumas alterações, geralmente pequenas, você pode converter facilmente um aplicativo existente para aproveitar os recursos do ILE. Depois de criar os módulos, combiná-los em alguns programas grandes pode ser a maneira mais fácil e menos dispendiosa de converter para ILE. v Tente limitar o número de programas de serviço que seu aplicativo usa.

Isso pode exigir que um programa de serviço seja criado a partir de mais de um módulo. As vantagens são um tempo de ativação mais rápido e um processo de ligação mais rápido.

| Não há respostas simples para o número de programas de serviço que um aplicativo deve usar. Se um | programa usa centenas de programas de serviço, provavelmente está usando muitos. Por outro lado, um | programa de serviço pode não ser prático também.

**84** Conceitos de ILE IBM i 7.1

**Capítulo 6. Gerenciamento do Grupo de Ativação**

Este capítulo contém exemplos de como estruturar um aplicativo usando grupos de ativação. Os tópicos incluem:

v Suporte a vários aplicativos v Usando o comando Recursos de Recuperação (RCLRSC) com programas OPM e ILE v Excluindo grupos de ativação com o comando Grupo de Ativação de Recuperação (RCLACTGRP) v Programas de serviço e grupos de ativação

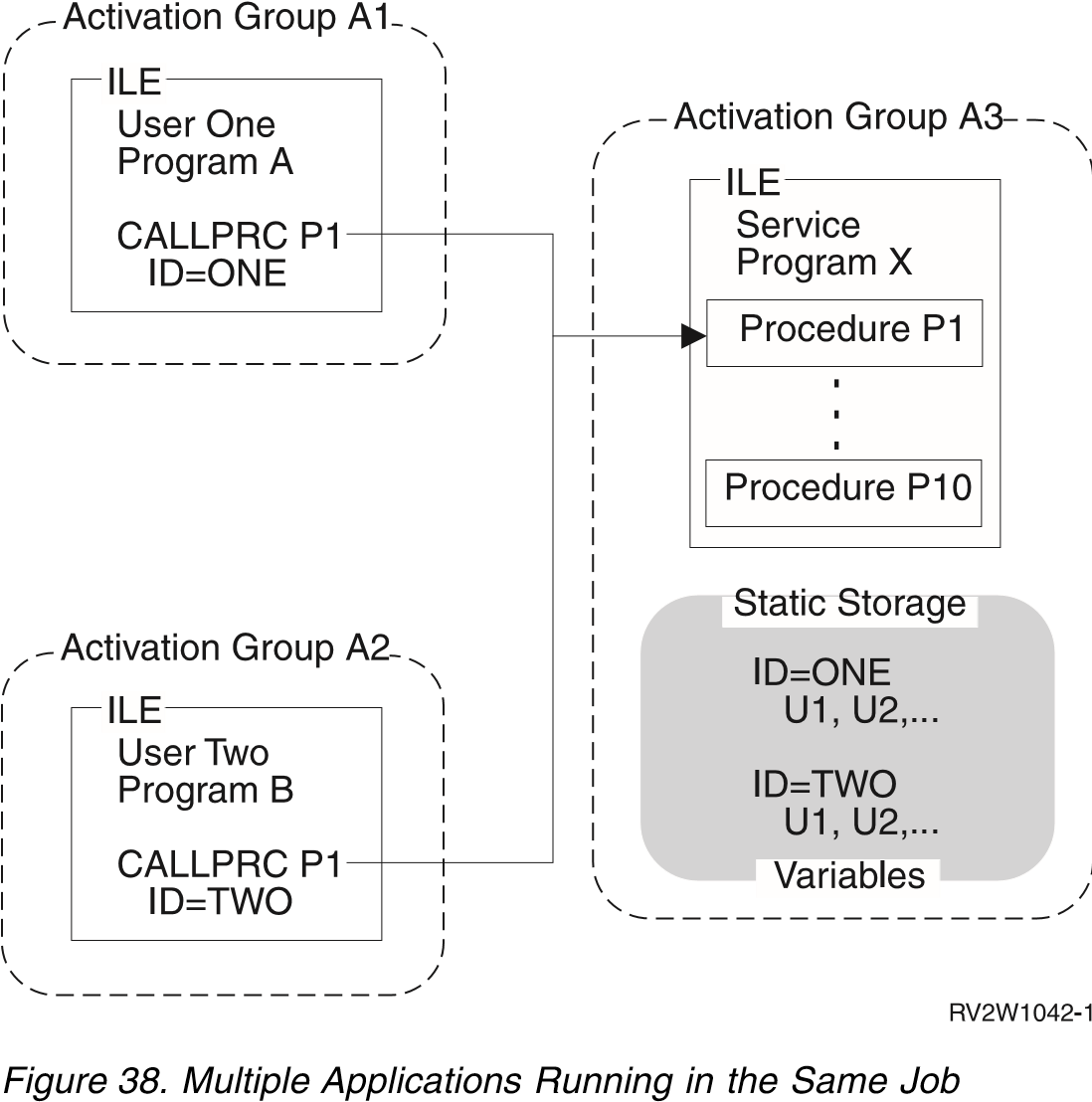
**Vários aplicativos em execução no mesmo trabalho**

Os grupos de ativação nomeados pelo usuário permitem que você deixe um grupo de ativação em um trabalho para uso posterior. Uma operação de retorno normal ou uma operação de pular (como longjmp() na ILE C) após o limite de controle não exclui o grupo de ativação.

Isso permite que você deixe seu aplicativo em seu último estado usado. Variáveis estáticas e arquivos abertos permanecem inalterados entre as chamadas em seu aplicativo. Isso pode economizar tempo de processamento e pode ser necessário para realizar a função que você está tentando fornecer.

No entanto, você deve estar preparado para aceitar solicitações de vários clientes independentes em execução no mesmo trabalho. O sistema não limita o número de programas ILE que podem ser vinculados ao seu programa de serviço ILE. Como resultado, talvez seja necessário oferecer suporte a vários clientes.

A Figura 38 mostra uma técnica que você pode usar para compartilhar funções de serviço comuns, mantendo as vantagens de desempenho de um grupo de ativação nomeado pelo usuário.



© Direitos autorais IBM Corp. 1997, 2010

Cada chamada para um procedimento no programa de serviço X requer um identificador de usuário. O ID do campo representa um identificador de usuário neste exemplo. Cada usuário é responsável por fornecer esse identificador. Você faz uma rotina de inicialização para retornar um identificador exclusivo para cada usuário.

Quando uma chamada é feita para o programa de serviço, o identificador de usuário é usado para localizar as variáveis de armazenamento relacionadas a esse usuário. Ao economizar tempo de criação do grupo de ativação, você pode oferecer suporte a vários clientes ao mesmo tempo.

**Comando Recuperar Recursos**

O comando Reclaim Resources (RCLRSC) depende de um conceito de sistema conhecido como **número de nível**. Um número de nível é um valor exclusivo atribuído pelo sistema a determinados recursos que você usa em um trabalho. Três números de nível são definidos da seguinte forma:

**Número do nível de chamada**

Cada entrada de pilha de chamadas recebe um número de nível exclusivo

**Número do nível de ativação do programa**

Cada ativação do programa OPM e ILE recebe um número de nível exclusivo

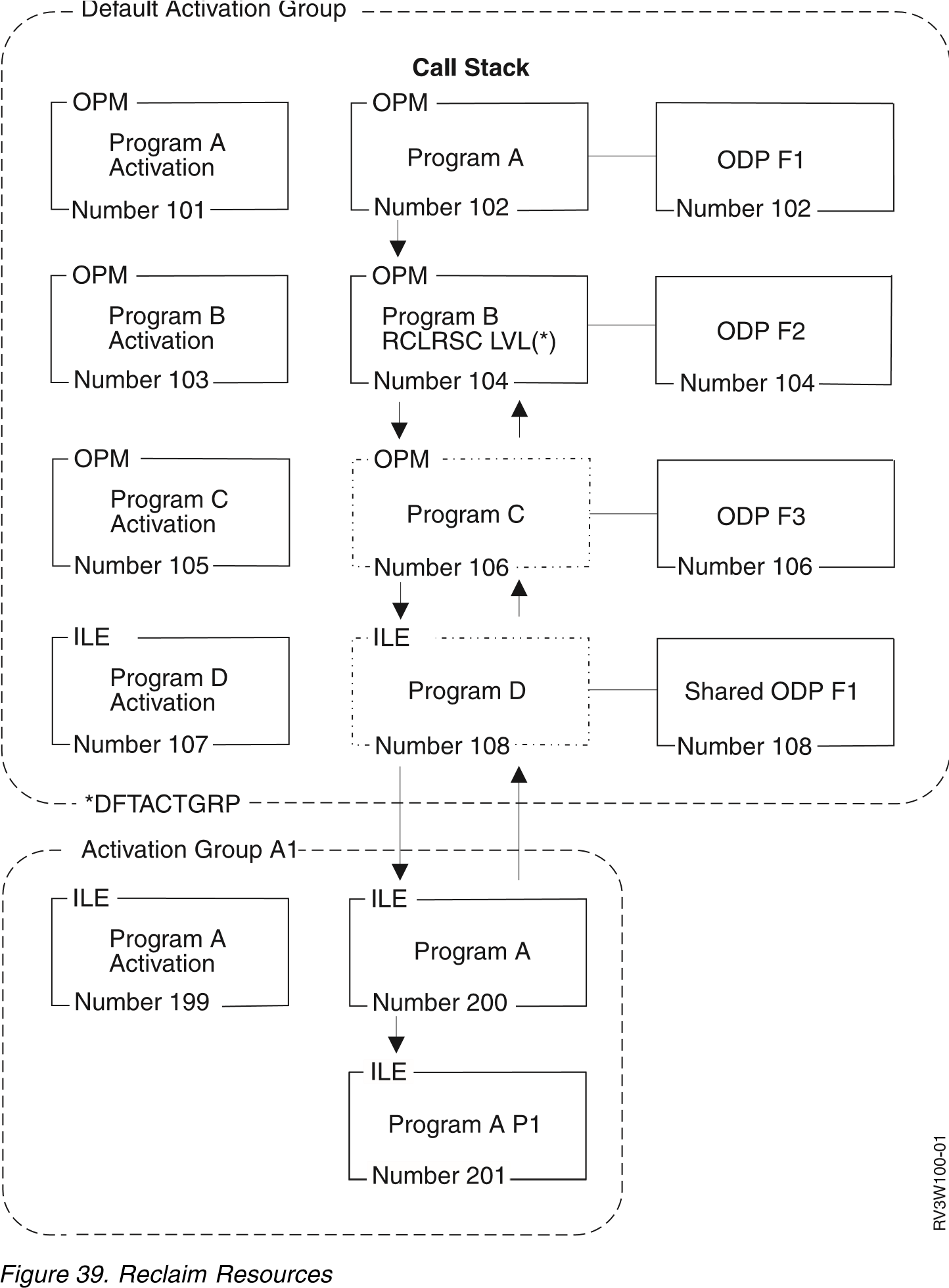
#### Número de nível do grupo de ativação

Cada grupo de ativação recebe um número de nível exclusivo

À medida que seu trabalho é executado, o sistema continua a atribuir números de nível exclusivos para cada nova ocorrência dos recursos que acabamos de descrever. Os números de nível são atribuídos em valor crescente. Recursos com números de nível mais alto são criados após recursos com números de nível mais baixos.

A Figura 39 na página 87 mostra um exemplo de uso do comando RCLRSC em programas OPM e ILE. O escopo em nível de chamada foi usado para os arquivos abertos mostrados neste exemplo. Quando o escopo em nível de chamada é usado, cada recurso de gerenciamento de dados recebe os mesmos números de nível que a entrada de pilha de chamadas que criou esse recurso.

**86** Conceitos de ILE IBM i 7.1



| Neste exemplo, a sequência de chamada é os programas A, B, C e D. Os programas D e C retornam ao programa B.

| O programa B está prestes a usar o comando RCLRSC com uma opção de LVL(\*). O comando RCLRSC usa

| o parâmetro level (LVL) para limpar recursos. Todos os recursos com um número de nível de chamada maior que o

| o número de nível de chamada da entrada da pilha de chamadas atual é limpo. Neste exemplo, o número de nível de chamada 104 é

| usado como ponto de partida. Todos os recursos maiores que o número de nível de chamada 104 são excluídos. Note que | os recursos no nível de chamada 200 e 201 não são afetados pelo RCLRSC porque estão em um grupo de ativação ILE. | RCLRSC funciona apenas nos grupos de ativação padrão.

Além disso, o armazenamento dos programas C e D e o caminho de dados abertos (ODP) para o arquivo F3 é fechado. O arquivo F1 é compartilhado com o ODP aberto no programa A. O ODP compartilhado é fechado, mas o arquivo F1 permanece aberto.

Capítulo 6. Gerenciamento do Grupo de Ativação

**Comando Recuperar Recursos para Programas OPM**

O comando Recursos de Recuperação (RCLRSC) pode ser usado para fechar arquivos abertos e liberar armazenamento estático para programas OPM que retornaram sem terminar. Algumas linguagens OPM, como RPG, permitem que você retorne sem terminar o programa. Se mais tarde você quiser fechar os arquivos do programa e liberar seu armazenamento, você pode usar o comando RCLRSC.

**Comando Recuperar Recursos para Programas ILE**

| Para programas ILE que são criados pelos comandos CRTBNDRPG e CRTBNDCL com

| DFTACTGRP(\*YES) especificado, o comando RCLRSC libera armazenamento estático, assim como faz para o OPM

| Programas. Para programas ILE que **não**  são criados pelos comandos CRTBNDRPG ou CRTBNDCL com

| DFTACTGRP(\*YES) especificado, o comando RCLRSC reinicializa todas as ativações que foram criadas

| em um grupo de ativação padrão, mas não libera o armazenamento estático. Programas ILE que usam grandes quantidades de

| o armazenamento estático deve ser ativado em um grupo de ativação ILE. A exclusão do grupo de ativação retorna isso

| armazenamento para o sistema. O comando RCLRSC fecha arquivos abertos por programas de serviço ou programas ILE | em execução em um grupo de ativação padrão. O comando RCLRSC não reinicializa o armazenamento estático de | e não afeta grupos de ativação não padrão.

Para usar o comando RCLRSC diretamente do ILE, você pode usar a API QCAPCMD ou um procedimento ILE CL. Usando a API QCAPCMD, você pode chamar diretamente comandos do sistema sem o uso de um programa CL. Na Figura 39 na página 87, chamar diretamente os comandos do sistema é importante porque talvez você queira usar o número de nível de chamada de um determinado procedimento ILE. Certas linguagens, como ILE C, também fornecem uma função do sistema que permite a execução direta de comandos do IBM i.

**Comando Recuperar Grupo de Ativação**

O comando Grupo de Ativação de Recuperação (RCLACTGRP) pode ser usado para excluir um grupo de ativação não padrão que não está em uso. Esse comando permite que as opções excluam todos os grupos de ativação qualificados ou excluam um grupo de ativação pelo nome.

**Programas de serviço e grupos de ativação**

Ao criar um programa de serviço ILE, decida se deseja especificar uma opção de \*CALLER ou um nome para o parâmetro ATTIGRP. Essa opção determina se o programa de serviço será ativado no grupo de ativação do chamador ou em um grupo de ativação nomeado separadamente. Qualquer escolha tem vantagens e desvantagens. Este tópico discute o que cada opção oferece.

Para a opção ACTGRP(\*CALLER), o programa de serviço funciona da seguinte maneira:

v As chamadas de procedimento estático são rápidas

As chamadas de procedimento estático no programa de serviço são otimizadas ao serem executadas no mesmo grupo de ativação.

v Dados externos compartilhados

Os programas de serviço podem exportar dados para serem usados por outros programas e programas de serviço no mesmo grupo de ativação.

v Recursos de gerenciamento de dados compartilhados

Arquivos abertos e outros recursos de gerenciamento de dados podem ser compartilhados entre o programa de serviço e outros programas no grupo de ativação. O programa de serviço pode emitir uma operação de confirmação ou uma operação de reversão que afeta os outros programas no grupo de ativação.

v Sem limite de controle

Exceções não tratadas dentro do programa de serviço percolate para os programas cliente. Os verbos de fim HLL usados no programa de serviço podem excluir o grupo de ativação dos programas cliente.

Para a opção ACTGRP(name), o programa de serviço funciona da seguinte maneira:

| v Separe o espaço de endereço para as variáveis, se estiver usando o modelo de armazenamento de nível único.

**88** Conceitos de ILE IBM i 7.1

| O programa cliente não pode manipular ponteiros para endereçar seu armazenamento de trabalho. Isso pode ser | importante se o seu programa de serviço estiver sendo executado com a autoridade adotada. v Recursos de gerenciamento de dados separados

Você tem seus próprios arquivos abertos e definições de compromisso. O compartilhamento acidental de arquivos abertos é impedido.

v Informações estatais controladas

Você controla quando o armazenamento do aplicativo é excluído. Usando verbos de fim HLL ou instruções de retorno de idioma normal, você pode decidir quando excluir o aplicativo. No entanto, você deve gerenciar as informações de estado para vários clientes.

**90 Conceitos** ILE IBM i 7.1

**Capítulo 7.**  **Chamadas para Procedimentos e Programas**

A pilha de chamadas ILE e os métodos de passagem de argumentos facilitam a comunicação entre idiomas, facilitando a escrita de aplicativos de linguagem mista. Este capítulo discute diferentes exemplos de chamadas dinâmicas de programa e chamadas de procedimento estático, que foram introduzidas em "Chamando um programa ou um procedimento" na página 18. Um terceiro tipo de chamada, a chamada de ponteiro de procedimento, é introduzido.

Além disso, este capítulo discute como oferecer suporte a interfaces de programação de aplicativos (APIs) OPM usando novas funções ILE ou conversões OPM para ILE.

**Pilha de chamadas**

A pilha de chamadas é uma lista de  **entradas** de  **pilha** de  **chamadas** de última entrada (LIFO), uma entrada para cada procedimento ou programa chamado. Cada entrada de pilha de chamadas tem informações sobre as variáveis automáticas para o procedimento ou programa e sobre outros recursos com escopo para a entrada de pilha de chamadas, como manipuladores de condição e manipuladores de cancelamento.

| Há uma pilha de chamadas por thread. Uma chamada adiciona uma nova entrada na pilha de chamadas para o procedimento chamado ou | e passa o controle para o objeto chamado. Um retorno remove a entrada da pilha e passa o controle de volta | para o procedimento de chamada ou programa. Consulte Threads no IBM i para obter mais informações.

**Exemplo** de **pilha**  de chamadas

A Figura 40 na página 92 contém um segmento de uma pilha de chamadas com dois programas: um programa OPM (Programa A) e um programa ILE (Programa B). O programa B contém três procedimentos: seu procedimento de entrada no programa, seu procedimento de entrada do usuário e outro procedimento (P1). Os conceitos de procedimento de entrada de programa (PEP) e procedimento de entrada de usuário (UEP) são definidos em "Objeto de módulo" na página 10. O fluxo de chamadas inclui as seguintes etapas:

1. Uma chamada de programa dinâmico para o Programa A.
2. O Programa A chama o Programa B, passando o controle para o seu PEP. Esta chamada para o Programa B é uma chamada de programa dinâmico.
3. O PEP chama a UEP. Esta é uma chamada de procedimento estático.
4. O UEP chama o procedimento P1. Esta é uma chamada de procedimento estático.

© Direitos autorais IBM Corp. 1997, 2010

Programa A

Módulo M1

Módulo M2

Programa B

**Pilha de chamadas**

OPM

ILE

RV2W1034-1

OPM

ILE

ILE

Dinâmico

Programa

Chamar

Estático

Procedimento

Chamar

Estático

Procedimento

Chamar

Entrada no Programa

Procedimento

Entrada no Programa

Procedimento

Procedimento P1

Procedimento P1

Entrada do usuário

Procedimento

Entrada do usuário

Procedimento

*Figura 40. Chamadas de programa* dinâmico  *e*  chamadas de procedimento *estático na pilha de chamadas*

| A Figura 40 ilustra a pilha de chamadas para este exemplo. A entrada chamada mais recentemente na pilha é representada | na parte inferior da pilha. É a entrada para o programa ou procedimento que está sendo processado no momento. O | O programa ou procedimento atual pode executar um dos seguintes procedimentos:

| v Chame outro procedimento ou programa, que adiciona outra entrada à parte inferior da pilha.

| v Retornar o controle para seu chamador depois que ele terminar o processamento, o que faz com que sua entrada de pilha de chamadas seja removida | da pilha.

| Suponha que, após o procedimento P1 ser feito, não é necessário mais processamento do Programa B. Procedimento P1

| retorna o controle para o UEP e a entrada para P1 é removida da pilha. Em seguida, o UEP retorna o controle

| para o PEP e a entrada UEP é removida da pilha. Finalmente, o PEP retorna o controle para o Programa A,

| e a entrada PEP é removida da pilha. Apenas a entrada do Programa A é deixada neste segmento da chamada | pilha. O programa A continua processando a partir do ponto em que fez a chamada dinâmica do programa para | Programa B.

**Chamadas para Programas e Chamadas para Procedimentos**

Três tipos de chamadas podem ser feitas durante o tempo de execução do ILE: chamadas dinâmicas de programa, chamadas de procedimento estático e chamadas de ponteiro de procedimento.

| Quando um programa ILE é ativado, todos os seus procedimentos, exceto seu PEP, tornam-se disponíveis para estática

| chamadas de procedimento e chamadas de ponteiro de procedimento. A ativação do programa ocorre quando o programa é chamado por um

| chamada de programa dinâmico e a ativação ainda não existe. Quando um programa é ativado, o serviço | programas que estão vinculados a este programa, para o qual a ativação diferida não se aplica, também são ativados. | Os procedimentos em um programa de serviço ILE podem ser acessados somente por chamadas de procedimento estático ou por procedimento | chamadas de ponteiro (não por chamadas dinâmicas de programa).

**Chamadas de procedimento**  estático

Uma chamada para um procedimento ILE adiciona uma nova entrada de pilha de chamadas à parte inferior da pilha e passa o controle para um procedimento especificado. Os exemplos incluem qualquer um dos seguintes:

1. Uma chamada para um procedimento no mesmo módulo
2. Uma chamada para um procedimento em um módulo diferente no mesmo programa ILE ou programa de serviço
3. Uma chamada para um procedimento que foi exportado de um programa de serviço ILE no mesmo grupo de ativação
4. Uma chamada para um procedimento que foi exportado de um programa de serviço ILE em um grupo de ativação diferente

| Nos exemplos 1, 2 e 3, a chamada de procedimento estático não cruza um limite de grupo de ativação. Esta chamada

| path é muito mais curto do que o caminho para uma chamada de programa dinâmico para um programa ILE ou OPM. No exemplo 4,

| a chamada cruza um limite de grupo de ativação e o processamento adicional é feito para alternar a ativação | recursos do grupo. O comprimento do caminho da chamada é maior do que o comprimento do caminho de uma chamada de procedimento estático dentro de um | grupo de ativação, mas ainda mais curto do que para uma chamada de programa dinâmico.

| Para uma chamada de procedimento estático, o procedimento chamado foi vinculado ao procedimento de chamada . A chamada sempre | acessa o mesmo procedimento. Por outro lado, o destino de uma chamada de ponteiro de procedimento pode variar com cada chamada.

**Chamadas** de ponteiro de procedimento

**As chamadas** de ponteiro de procedimento fornecem uma maneira de chamar um procedimento dinamicamente. Por exemplo, manipulando matrizes, ou tabelas, de nomes ou endereços de procedimento, você pode rotear dinamicamente uma chamada de procedimento para procedimentos diferentes.

As chamadas de ponteiro de procedimento adicionam entradas à pilha de chamadas exatamente da mesma maneira que as chamadas de procedimento estático. Qualquer procedimento que possa ser chamado usando uma chamada de procedimento estático também pode ser chamado por meio de um ponteiro de procedimento. Se o procedimento chamado estiver no mesmo grupo de ativação, o custo de uma chamada de ponteiro de procedimento será quase idêntico ao custo de uma chamada de procedimento estático.

**Passando argumentos para procedimentos ILE**

Em uma chamada de procedimento ILE, um  **argumento** é uma expressão que representa um valor que o procedimento de chamada passa para o procedimento especificado na chamada . As linguagens ILE usam três métodos para passar argumentos:

**por valor, diretamente**

O valor do objeto de dados é colocado diretamente na lista de argumentos.

#### por valor, indiretamente

O valor do objeto de dados é copiado para um local temporário. O endereço da cópia (um ponteiro) é colocado na lista de argumentos.

#### por referência

Um ponteiro para o objeto de dados é colocado na lista de argumentos. As alterações feitas pelo procedimento chamado no argumento são refletidas no procedimento de chamada.

A Figura 41 na página 94 ilustra esses estilos de passagem de argumento. Nem todas as linguagens ILE suportam a passagem por valor, diretamente. Os estilos de passagem disponíveis são descritos nos guias do programador ILE HLL .

Por valor, diretamente

uma cópia do argumento

Por valor, indiretamente

ponteiro

uma cópia do argumento

Por referência

ponteiro

o argumento real

RV2W1027-1

*Figura 41. Métodos*  para *passar argumentos para procedimentos ILE*

A semântica HLL geralmente determina quando os dados são passados por valor e quando são passados por referência . Por exemplo, o ILE C passa e aceita argumentos por valor, diretamente, enquanto para ILE COBOL e ILE RPG , os argumentos geralmente são passados por referência. Você deve garantir que o programa ou procedimento de chamada passe argumentos da maneira esperada pelo procedimento chamado. Os guias do programador ILE HLL contêm mais informações sobre como passar argumentos para diferentes linguagens.

Um máximo de 400 argumentos são permitidos em uma chamada de procedimento estático. Cada linguagem ILE pode restringir ainda mais o número máximo de argumentos. As linguagens ILE oferecem suporte aos seguintes estilos de passagem de argumentos:

v ILE C passa e aceita argumentos por valor diretamente, ampliando inteiros e valores de ponto flutuante por padrão. Os argumentos podem ser passados sem ampliação ou por valor indiretamente se você especificar os valores apropriados na diretiva de argumento #pragma para a função chamada.

v ILE C++ passa e aceita argumentos por valor diretamente. O C++ não amplia parâmetros e valores de ponto flutuante por padrão. Os argumentos podem ser ampliados ou passados por valor indiretamente se você especificar os valores apropriados no especificador de ligação externa para a declaração da função chamada.

v ILE COBOL passa e aceita argumentos por valor, por referência ou por valor indiretamente. Os parâmetros que são passados por valor não são ampliados.

v ILE RPG passa e aceita argumentos por valor ou por referência. O RPG não amplia inteiros e valores de ponto flutuante por padrão, mas isso está disponível para parâmetros passados por valor, por codificação

EXTPROC(\*CWIDEN).

v ILE CL passa e aceita argumentos por referência e por valor. Os parâmetros que são passados por valor não são ampliados.

#### Resultados da função

Para oferecer suporte a HLLs que permitem a definição de funções (procedimentos que retornam um argumento de resultado), o modelo pressupõe que um argumento de resultado de função especial pode estar presente, conforme mostrado em Figura 42 na página 95. Conforme descrito nos guias do programador ILE HLL, as linguagens ILE que oferecem suporte a resultados de função usam um mecanismo comum para retornar resultados de função.

Procedimento de chamada Chamado procedimento passa Argumentos Maio retornar um função resultado

Vocação

Procedimento

Procedimento chamado

Retornar

Chamar

RV2W1028-1

*Figura 42. Terminologia* do  *argumento de chamada* do *programa*

#### Argumentos omitidos

Todas as linguagens ILE podem simular argumentos omitidos, o que permite o uso do mecanismo de código de feedback para manipuladores de condição ILE e outros procedimentos de tempo de execução. Por exemplo, se um procedimento ILE C ou uma API vinculável ao ILE estiver esperando um argumento passado por referência, às vezes você poderá omitir o argumento passando um null ponteiro em seu lugar. Para obter informações sobre como especificar um argumento omitido em uma linguagem ILE específica, consulte o guia do programador para essa linguagem. O tópico API da categoria Programação do Centro de Informações do IBM i especifica quais argumentos podem ser omitidos para cada API.

Para linguagens ILE que não fornecem uma maneira intrínseca para um procedimento chamado testar se um argumento foi omitido, a API vinculável Test for Omitted Argument (CEETSTA) está disponível .

**Chamadas dinâmicas de programas**

Uma chamada de programa dinâmico é uma chamada feita para um objeto de programa . Por exemplo, quando você usa o comando CL CALL, você está fazendo uma chamada de programa dinâmico.

Os programas OPM são chamados usando chamadas dinâmicas de programas. Os programas OPM também são limitados a fazer apenas chamadas dinâmicas de programas.

Os programas ILE também são chamados por chamadas dinâmicas de programas. Os procedimentos dentro de um programa ILE ativado podem ser acessados usando chamadas de procedimento estático ou chamadas de ponteiro de procedimento.

Em contraste com as chamadas de procedimento estático, que são vinculadas em tempo de compilação, os símbolos para chamadas de programa dinâmico são resolvidos para endereços quando a chamada é executada . Como resultado, uma chamada de programa dinâmico usa mais recursos do sistema do que uma chamada de procedimento estático. Exemplos de uma chamada de programa dinâmico incluem:

v Uma chamada para um programa ILE ou um programa OPM v Uma chamada para uma API não vinculável

Uma chamada de programa dinâmico para um programa ILE passa o controle para o PEP do programa identificado, que então passa o controle para o UEP do programa. Depois que o programa chamado é feito o processamento, o controle é passado de volta para a instrução após a instrução do programa de chamada.

**Passando argumentos em uma**  chamada **de programa dinâmico**

| As chamadas para programas ILE ou OPM (em contraste com as chamadas para procedimentos ILE) passam argumentos por referência | o que significa que o programa chamado recebe o endereço de cada argumento.

Ao usar uma chamada de programa dinâmico, você precisa saber o método de passagem de argumento que é esperado pelo programa chamado e como simulá-lo , se necessário. Um máximo de 255 argumentos são permitidos em uma chamada de programa dinâmico. Cada linguagem ILE pode restringir ainda mais o número máximo de argumentos. Algumas linguagens ILE suportam a função interna CALLPGMV, que permite um máximo de 16383 argumentos. Informações sobre como usar os diferentes métodos de passagem estão contidas nos guias do programador ILE HLL .

**Compatibilidade de dados entre idiomas**

As chamadas ILE permitem que os argumentos sejam passados entre procedimentos escritos em diferentes HLLs. Para facilitar o compartilhamento de dados entre os HLLs, algumas linguagens ILE adicionaram tipos de dados. Por exemplo, o ILE COBOL adicionou USAGE PROCEDURE-POINTER como um novo tipo de dados.

Para passar argumentos entre HLLs, você precisa saber o formato que cada HLL espera dos argumentos que está recebendo. O procedimento de chamada é necessário para garantir que os argumentos tenham o tamanho e o tipo esperados pelo procedimento chamado. Por exemplo, uma função ILE C pode esperar um inteiro de 4 bytes, mesmo que um inteiro curto (2 bytes) seja declarado na lista de parâmetros. Informações sobre como corresponder aos requisitos de tipo de dados para passar argumentos estão contidas nos guias do programador ILE HLL .

**Sintaxe para passar argumentos em**  aplicativos **de linguagem mista**

Algumas linguagens ILE fornecem sintaxe para passar argumentos para procedimentos em outras linguagens ILE. Por exemplo, o ILE C fornece um argumento #pragma para passar argumentos de valor para outros procedimentos ILE por valor indiretamente ; RPG tem valores especiais para a palavra-chave protótipo EXTPROC.

**Descritores** Operacionais

Descritores operacionais podem ser úteis para você se você estiver escrevendo um procedimento ou API que possa receber argumentos de procedimentos escritos em diferentes HLLs.  **Os descritores** **operacionais** fornecem informações descritivas para o procedimento chamado nos casos em que o procedimento chamado não pode antecipar com precisão a forma do argumento (por exemplo, diferentes tipos de strings). As informações adicionais permitem que o procedimento interprete adequadamente os argumentos.

O argumento fornece o valor; o descritor operacional fornece informações sobre o tamanho e o tipo do argumento. Por exemplo, essas informações podem incluir o comprimento de uma cadeia de caracteres e o tipo de cadeia de caracteres.

Com os descritores operacionais, serviços como APIs vinculáveis não precisam ter uma variedade de ligações diferentes para cada HLL, e os HLLs não precisam imitar tipos de dados incompatíveis . Algumas APIs vinculáveis de ILE usam descritores operacionais para acomodar a falta de tipos de dados de cadeia de caracteres comuns entre HLLs. A presença do descritor operacional é transparente para o usuário da API.

Os descritores operacionais suportam a semântica HLL enquanto são invisíveis para procedimentos que não os usam ou esperam. Cada linguagem ILE pode usar tipos de dados apropriados à linguagem. Cada compilador de linguagem ILE fornece pelo menos um método para gerar descritores operacionais. Para obter mais informações sobre a semântica HLL para descritores operacionais, consulte o manual de referência do ILE HLL .

Os descritores operacionais são distintos de outros descritores de dados com os quais você pode estar familiarizado. Por exemplo, eles não estão relacionados aos descritores associados a dados ou arquivos distribuídos.

#### Requisitos dos Descritores Operacionais

Você precisa usar descritores operacionais quando eles são esperados por um procedimento chamado escrito em uma linguagem ILE diferente e quando eles são esperados por uma API vinculável ILE. Geralmente, as APIs vinculáveis exigem descritores para a maioria dos argumentos de cadeia de caracteres. As informações sobre APIs vinculáveis no tópico API da categoria Programação do Centro de Informações do IBM i especificam se uma determinada API vinculável requer descritores operacionais.

#### Ausência de um descritor necessário

| A omissão de um descritor necessário é um erro. Se um procedimento exigir um descritor para um | específico , esse requisito faz parte da interface desse procedimento. Se um descritor necessário não for | desde que, o procedimento chamado falhará em tempo de execução.

#### Presença de um descritor desnecessário

A presença de um descritor que não é necessário não interfere no acesso do procedimento chamado aos argumentos. Se um descritor operacional não for necessário ou esperado, o procedimento chamado simplesmente o ignorará.

| **Nota:** No entanto, a geração de descritores desnecessários pode diminuir o desempenho, porque o uso do descritor | efetivamente aumenta o comprimento do caminho de chamada.

#### APIs vinculáveis para acesso ao descritor operacional

Os descritores são normalmente acessados diretamente por um procedimento chamado de acordo com a semântica do HLL no qual o procedimento é escrito. Uma vez que um procedimento é programado para esperar descritores operacionais, nenhum tratamento adicional é geralmente exigido pelo programador. No entanto, às vezes, um procedimento chamado precisa determinar se os descritores que ele requer estão presentes antes de acessá-los. Para este propósito, as seguintes APIs vinculáveis são fornecidas:

v Recuperar informações do descritor operacional (CEEDOD) API vinculável v Obter informações de cadeia de caracteres (CEEGSI) API vinculável

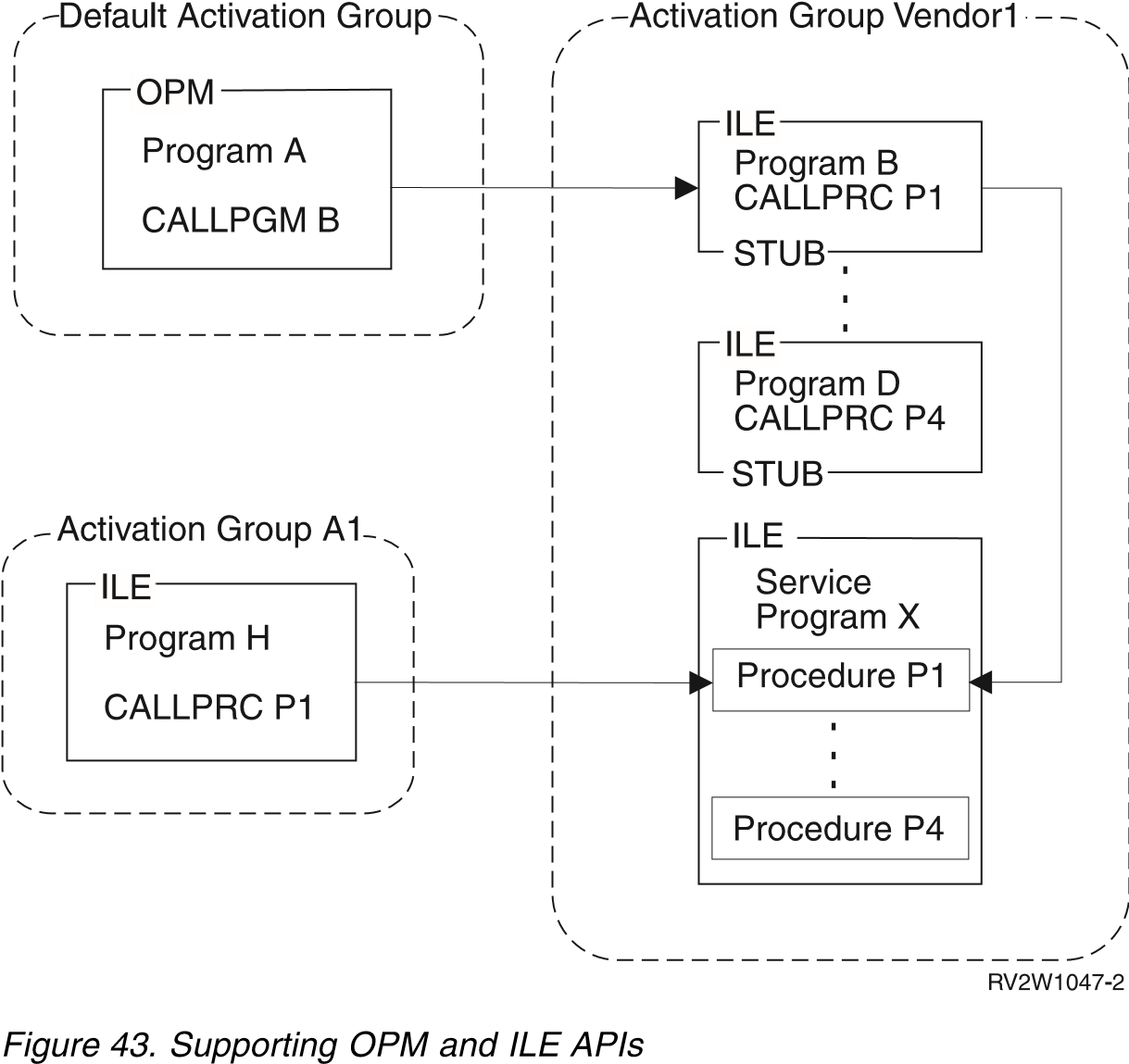
**Suporte para APIs OPM e ILE**

Ao desenvolver novas funções no ILE ou converter um aplicativo existente em ILE, convém continuar a oferecer suporte a APIs de nível de chamada do OPM. Este tópico explica uma técnica que pode ser usada para realizar esse suporte duplo enquanto mantém seu aplicativo no ILE.

| Os programas de serviço ILE fornecem uma maneira de desenvolver e fornecer APIs vinculáveis que podem ser acessadas | de todos os idiomas ILE. Para fornecer as mesmas funções aos programas OPM, você precisa considerar o fato de | que um procedimento em um programa de serviço ILE não pode ser chamado diretamente de um programa OPM.

A técnica a ser usada é desenvolver stubs de programa ILE para cada API vinculável que você planeja suportar. Talvez você queira nomear as APIs vinculáveis da mesma forma que os stubs do programa ILE ou escolher nomes diferentes. Cada stub de programa ILE contém uma chamada de procedimento estático para a API vinculável real.

Um exemplo dessa técnica é mostrado na Figura 43 na página 98.



Os programas B a D são os esboços do programa ILE. O programa de serviço X contém a implementação real de cada API vinculável. Cada stub de programa e o programa de serviço recebem o mesmo nome de grupo de ativação. Neste exemplo, o nome do grupo de ativação VENDOR1 é escolhido.

O grupo de ativação VENDOR1 é criado pelo sistema quando necessário. A chamada dinâmica do programa OPM A cria o grupo de ativação na primeira chamada de um programa OPM . A chamada de procedimento estático do programa ILE H cria o grupo de ativação quando o programa ILE H é ativado. Uma vez que o grupo de ativação existe, ele pode ser usado a partir do programa A ou do programa H.

| Escreva a implementação de sua API em um procedimento ILE (procedimento P1 neste exemplo). Este

| procedimento pode ser chamado diretamente através de uma chamada de procedimento ou indiretamente através de um programa dinâmico

| chamada para o stub do programa. Você deve ter cuidado ao executar ações que dependem da pilha de chamadas

| estrutura, como o envio de mensagens de exceção. Um retorno normal do stub do programa ou do

| o procedimento de implementação deixa o grupo de ativação no trabalho para uso posterior. Você pode implementar sua API

| com o conhecimento de que um limite de controle é estabelecido para o stub do programa ou para o | procedimento de execução em cada chamada. Os verbos finais HLL excluem o grupo de ativação se a chamada | originado de um programa OPM ou um programa ILE.

**Capítulo 8.**   **Gerenciamento** **de armazenamento**

O sistema operacional fornece suporte de armazenamento para os idiomas de alto nível ILE. Esse suporte de armazenamento elimina a necessidade de gerenciadores de armazenamento exclusivos para o ambiente de tempo de execução de cada linguagem. Ele evita incompatibilidades entre diferentes gerenciadores de armazenamento e mecanismos em idiomas de alto nível.

O sistema operacional fornece o armazenamento automático, estático e dinâmico usado por programas e procedimentos em tempo de execução. O armazenamento automático e estático é gerenciado pelo sistema operacional. Ou seja, a necessidade de armazenamento automático e estático é conhecida em tempo de compilação a partir de declarações de variáveis de programa. O armazenamento dinâmico é gerenciado pelo programa ou procedimento. A necessidade de armazenamento dinâmico é conhecida apenas em tempo de execução.

Quando a ativação do programa ocorre, o armazenamento estático para variáveis de programa é alocado e inicializado.

Quando um programa ou procedimento começa a ser executado, o armazenamento automático é alocado. A pilha de armazenamento automático é estendida para variáveis à medida que o programa ou procedimento é adicionado à pilha de chamadas.

À medida que um programa ou procedimento é executado, o armazenamento dinâmico é alocado sob o controle do programa. Esse armazenamento é estendido à medida que o armazenamento adicional é necessário. Você tem a capacidade de controlar o armazenamento dinâmico. O restante deste capítulo concentra-se no armazenamento dinâmico e nas maneiras pelas quais ele pode ser controlado.

**Pilha de armazenamento de nível**  único

Um **heap** é uma área de armazenamento que é usada para alocações de armazenamento dinâmico. A quantidade de armazenamento dinâmico que é exigida por um aplicativo depende dos dados que estão sendo processados pelo programa e procedimentos que usam um heap . O sistema operacional permite o uso de vários heaps de armazenamento de nível único que são criados e descartados dinamicamente. A instrução ALCHSS sempre usa armazenamento de nível único. Alguns idiomas também oferecem suporte ao uso de teraspace para armazenamento dinâmico.

**Características da pilha**

Cada pilha tem as seguintes características:

v O sistema atribui um identificador de pilha exclusivo a cada pilha dentro do grupo de ativação.

O identificador de pilha para o heap padrão é sempre zero.

Uma API vinculável ao gerenciamento de armazenamento, chamada por um programa ou procedimento, usa o identificador de pilha para identificar a pilha na qual ela deve agir. A API vinculável deve ser executada dentro do grupo de ativação que possui o heap. v O grupo de ativação que cria um heap também o possui.

Como os grupos de ativação possuem heaps, o tempo de vida de um heap não é maior do que o do grupo de ativação proprietário. O identificador de heap é significativo e exclusivo somente dentro do grupo de ativação que o possui. v O tamanho de um heap é estendido dinamicamente para satisfazer solicitações de alocação.

O tamanho máximo do heap é de 4 gigabytes menos 512K bytes. Este é o tamanho máximo do heap se o número total de alocações (a qualquer momento) não exceder 128.000. v O tamanho máximo de qualquer alocação única de um heap é limitado a 16 megabytes menos 64K bytes.

**Pilha** padrão

A primeira solicitação de armazenamento dinâmico do heap padrão dentro de um grupo de ativação que está usando armazenamento de nível único resulta na criação de um heap padrão do qual o armazenamento a alocação ocorre. Se não houver armazenamento suficiente no heap para atender a quaisquer solicitações subsequentes de armazenamento dinâmico, o sistema estenderá o heap e alocará armazenamento adicional.

© Direitos autorais IBM Corp. 1997, 2010

O armazenamento dinâmico alocado permanece alocado até que seja explicitamente liberado ou até que o sistema descarte o heap. O heap padrão é descartado somente quando o grupo de ativação proprietário termina.

Os programas no mesmo grupo de ativação compartilham automaticamente o armazenamento dinâmico, desde que o heap padrão aloque o armazenamento. No entanto, você pode isolar o armazenamento dinâmico que é usado por alguns programas e procedimentos dentro de um grupo de ativação. Você faz isso criando um ou mais heaps.

**Heaps criados pelo** usuário

Você pode criar e descartar explicitamente um ou mais heaps usando APIs vinculáveis ILE. Isso lhe dá a capacidade de gerenciar os heaps e o armazenamento dinâmico que é alocado a partir desses heaps.

Por exemplo, o sistema pode ou não compartilhar armazenamento dinâmico alocado em heaps criados pelo usuário para programas dentro de um grupo de ativação. O compartilhamento de armazenamento dinâmico depende do identificador de pilha que é referido pelos programas. Você pode usar mais de um heap para evitar o compartilhamento automático de armazenamento dinâmico. Dessa forma, você pode isolar grupos lógicos de dados. A seguir estão alguns motivos adicionais para usar um ou mais heaps criados pelo usuário:

v Você pode agrupar determinados objetos de armazenamento para atender a um requisito único. Depois de atender a esse requisito, você pode liberar o armazenamento dinâmico que foi alocado por uma única chamada para a API vinculável Descartar Heap (CEEDSHP ). Essa operação libera o armazenamento dinâmico e descarta o heap. Dessa forma, o armazenamento dinâmico está disponível para atender a outras solicitações.

v Você pode liberar vários armazenamentos dinâmicos alocados de uma só vez usando as APIs vinculáveis Mark Heap (CEEMKHP) e Release Heap (CEERLHP ). A API vinculável CEEMKHP permite que você marque um heap. Quando estiver pronto para liberar o grupo de alocações que foram feitas desde que o heap foi marcado, use a API vinculável CEERLHP . O uso das funções de marcação e liberação deixa o heap intacto, mas libera o armazenamento dinâmico alocado a partir dele. Dessa forma, você pode evitar a sobrecarga do sistema associada à criação de heap reutilizando heaps existentes para atender aos requisitos de armazenamento dinâmico.

v Seus requisitos de armazenamento podem não corresponder aos atributos de armazenamento que definem o heap padrão. Por exemplo, o tamanho inicial do heap padrão é de 4K bytes. No entanto, você precisa de várias alocações de armazenamento dinâmico que, juntas, excedam 4 K bytes. Você pode criar um heap com um tamanho inicial maior do que 4K bytes. Isso reduz a sobrecarga do sistema que, de outra forma, ocorreria tanto ao estender implicitamente o heap quanto subsequentemente acessar as extensões do heap. Da mesma forma, você pode ter extensões de pilha maiores que 4K bytes. Para obter informações sobre como definir tamanhos de heap, consulte "Estratégia de alocação de heap" na página 101 e a discussão sobre atributos de heap.

Você pode ter outros motivos para usar vários heaps em vez do heap padrão. As APIs vinculáveis ao gerenciamento de armazenamento oferecem a capacidade de gerenciar os heaps que você cria e o armazenamento dinâmico alocado nesses heaps. Para obter informações sobre as APIs de gerenciamento de armazenamento, consulte a coleção de tópicos de API na categoria Programação do Centro de Informações do IBM i.

**Suporte** a **heap** único

Os idiomas que não têm suporte intrínseco ao armazenamento de vários heap usam o heap de armazenamento de nível único padrão. Não é possível usar as APIs vinculáveis Descartar Heap (CEEDSHP), Mark Heap (CEEMKHP) ou Release Heap (CEERLHP) com o heap padrão. Você pode liberar o armazenamento dinâmico alocado pelo heap padrão usando operações livres explícitas ou encerrando o grupo de ativação que o possui.

Essas restrições ao uso do heap padrão ajudam a impedir a liberação inadvertida do armazenamento dinâmico alocado em aplicativos de idioma misto. Lembre-se de considerar as operações de heap de liberação e descarte de heap como inseguras para aplicativos grandes que reutilizam o código existente com suporte de armazenamento potencialmente diferente. Lembre-se de não usar operações de pilha de liberação que sejam válidas para a pilha padrão. Isso faz com que várias partes de um aplicativo que usa a função de marca corretamente quando usadas separadamente possivelmente falhem quando usadas juntas.

Conceitos de ILE IBM i 7.1

**Estratégia** de **alocação de pilha**

Os atributos associados ao heap padrão são definidos pelo sistema por meio de uma estratégia de alocação padrão. Essa estratégia de alocação define atributos como um tamanho de criação de pilha de 4K bytes e um tamanho de extensão de 4K bytes. Não é possível alterar essa estratégia de alocação padrão.

No entanto, você pode controlar os heaps que você cria explicitamente por meio da API vinculável Create a Heap (CEECRHP ). Você também pode definir uma estratégia de alocação para heaps criados explicitamente por meio da API vinculável Define Heap Allocation Strategy (CEE4DAS ). Em seguida, quando você cria explicitamente um heap, os atributos de heap são fornecidos pela estratégia de alocação que você definiu. Dessa forma, você pode definir estratégias de alocação separadas para um ou mais heaps criados explicitamente.

Você pode usar a API vinculável do CEECRHP sem definir uma estratégia de alocação. Nesse caso, o heap é definido pelos atributos do tipo de estratégia de alocação \_CEE4ALC. O tipo de estratégia de alocação \_CEE4ALC especifica um tamanho de criação de pilha de 4K bytes e um tamanho de extensão de 4K bytes. O tipo de estratégia de alocação \_CEE4ALC contém os seguintes atributos:

|  |  |  |
| --- | --- | --- |
| Max\_Sngl\_Alloc = 16MB - 64K | | /\* tamanho máximo de uma única alocação \*/ |
| Min\_Bdy | = 16 | /\* alinhamento mínimo dos limites de qualquer atribuição \*/ |
| Crt\_Size | = 4K | /\* tamanho de criação inicial da pilha \*/ |
| Ext\_Size | = 4K | /\* o tamanho da extensão da pilha \*/ |
| Alloc\_Strat | = 0 | /\* uma escolha para a estratégia de atribuição \*/ |
| No\_Mark | = 1 | /\* uma opção de desafectação de grupos \*/ |
| Blk\_Xfer | = 0 | /\* uma opção para transferência em bloco de uma pilha \*/ |
| PAG | = 0 | /\* uma opção para a criação de heap em um PAG \*/ |
| Alloc\_Init | = 0 | /\* uma opção para inicialização de alocação \*/ |
| Init\_Value | = 0x00 | /\* valor de inicialização \*/ |

Os atributos mostrados aqui ilustram a estrutura do tipo de estratégia de alocação de \_CEE4ALC. Para obter informações sobre todos os atributos de estratégia de alocação de \_CEE4ALC, consulte a coleção de tópicos da API na categoria Programação do Centro de Informações do IBM i.

**Interfaces** de  **pilha de armazenamento de** **nível** único

APIs vinculáveis são fornecidas para todas as operações de heap. Os aplicativos podem ser escritos usando as APIs vinculáveis, funções intrínsecas à linguagem ou ambas.

As APIs vinculáveis se enquadram nas seguintes categorias:

v Operações básicas de heap. Essas operações podem ser usadas no heap padrão e em heaps criados pelo usuário.

A API vinculável de armazenamento livre (CEEFRST) libera uma alocação anterior de armazenamento de pilha.

A API vinculável Get Heap Storage (CEEGTST) aloca o armazenamento dentro de um heap.

A API vinculável Reallocate Storage (CEECZST) altera o tamanho do armazenamento alocado anteriormente.

v Operações de pilha estendidas. Essas operações podem ser usadas somente em heaps criados pelo usuário.

A API vinculável Create Heap (CEECRHP) cria um novo heap.

A API vinculável Descartar Heap (CEEDSHP) descarta um heap existente.

A API vinculável Mark Heap (CEEMKHP) retorna um token que pode ser usado para identificar o armazenamento de heap a ser liberado pela API vinculável CEERLHP .

A API vinculável de pilhagem de liberação (CEERLHP) libera todo o armazenamento alocado no heap desde que a marca foi especificada.

v Estratégias de alocação de heap

A API vinculável Define Heap Allocation Strategy (CEE4DAS) define uma estratégia de alocação que determina os atributos de um heap criado com a API vinculável CEECRHP.

Para obter informações sobre as APIs de gerenciamento de armazenamento, consulte a coleção de tópicos de API na categoria Programação do Centro de Informações do IBM i.

Capítulo 8. Gerenciamento de armazenamento

# Suporte a pilha

Por padrão, o armazenamento dinâmico fornecido por malloc, calloc, realloc e new é o mesmo tipo de armazenamento que o modelo de armazenamento do programa raiz no grupo de ativação . No entanto, quando o modelo de armazenamento de nível único está em uso, o armazenamento teraspace é fornecido por essas interfaces se a opção de compilador TERASPACE(\*YES \*TSIFC) foi especificada. Da mesma forma, um programa de modelo de armazenamento de nível único pode usar explicitamente APIs vinculáveis para trabalhar com teraspace, como \_C\_TS\_malloc, \_C\_TS\_free, \_C\_TS\_realloc e \_ C\_TS\_calloc.

Para obter detalhes sobre como você pode usar o armazenamento teraspace, consulte o Capítulo 4, "Teraspace e armazenamento de nível único", na página 45.

Se você optar por usar as APIs vinculáveis de gerenciamento de armazenamento CEExxxx e as funções ILE C malloc(), calloc(), realloc() e free(), as regras a seguir aplicar:

v O armazenamento dinâmico alocado através das funções C malloc(), calloc() e realloc( ), não pode ser liberado ou realocado com as APIs vinculáveis CEEFRST e CEECZST .

v O armazenamento dinâmico alocado pela API vinculável CEEGTST pode ser liberado com a função free().

v O armazenamento dinâmico inicialmente alocado com a API vinculável CEEGTST pode ser realocado com a função realloc().

Outras linguagens, como COBOL, não têm modelo de armazenamento de pilha. Essas linguagens podem acessar o modelo de armazenamento dinâmico ILE por meio das APIs vinculáveis para armazenamento dinâmico.

| RPG tem códigos de operação ALLOC, REALLOC e DEALLOC, e funções internas %ALLOC e

| %REALLOC para acessar o armazenamento de pilha. Um módulo RPG pode usar armazenamento de pilha de nível único ou

| armazenamento de pilha teraspace. Para módulos com modelo de armazenamento teraspace, o tipo padrão de armazenamento de pilha é

| teraespaço. Para módulos com modelo de armazenamento herdado ou de nível único, o tipo padrão de armazenamento de pilha é

| nível único. No entanto, você pode definir explicitamente o tipo de armazenamento de pilha usando a palavra-chave ALLOC no

| Especificação de controle. O suporte a RPG usa as APIs vinculáveis CEEGTST, CEECZST e CEEFRST para | operações de armazenamento de pilha de nível único e usa as C\_TS\_ \_C\_TS\_realloc malloc(), \_C\_TS\_free() e \_|() funções para operações de armazenamento de pilha teraspace.

# Armazenamento local de thread

Os compiladores ILE C, ILE C++ e ILE RPG oferecem suporte a TLS (armazenamento local de thread ). As variáveis TLS para cada programa ou programa de serviço são organizadas em um quadro TLS. O quadro TLS contém uma cópia inicializada de cada variável TLS associada ao programa ou  programa de serviço. Uma cópia do quadro TLS é criada para cada thread que executa o programa ou programa de serviço. Para obter informações sobre o suporte disponível em um compilador específico, consulte a documentação da linguagem de alto nível (HLL) específica.

Uma variável TLS é semelhante a uma variável estática, exceto que existe uma cópia exclusiva da variável TLS para cada thread. Consulte a tabela a seguir para obter as diferenças entre variáveis TLS e variáveis estáticas. *Tabela 9.*  *Diferenças entre* *variáveis TLS e variáveis* *estáticas*

|  |  |  |
| --- | --- | --- |
|  | **Variável** estática | **Variável TLS** |
| Quando o armazenamento para a variável é alocado? | Quando o programa ou programa de serviço é ativado. | Quando o fio toca pela primeira vez o  Quadro TLS que contém a variável. |
| Quando a variável é inicializada? | Quando o programa ou programa de serviço é ativado. algarismo | Quando o fio toca pela primeira vez o  Quadro TLS que contém a variável. |
| Quando o armazenamento da variável é liberado? | Quando o programa ou programa de serviço é desativado. | Quando o fio é destruído. |
| Cada thread tem sua própria cópia da variável? | Não, uma única cópia é compartilhada por todos os threads. | Sim. |

Conceitos de ILE IBM i 7.1

*Tabela 9.*  *Diferenças entre* variáveis  *TLS e variáveis* *estáticas (continuação)*

|  |  |  |
| --- | --- | --- |
|  | **Variável** estática | **Variável TLS** |
| A variável é armazenada em armazenamento de nível único ou armazenamento teraspace? | Depende do grupo de ativação do programa ou programa de serviço. 1 | As variáveis TLS são sempre armazenadas no armazenamento teraspace. 1 |
| 1  Consulte o Capítulo 4, "Teraspace e armazenamento de nível único", na página 45 para obter mais informações. algarismo  Isso representa o momento em que a variável é inicializada diretamente pelo sistema. A variável pode ser inicializada indiretamente pelo seu HLL posteriormente . | | |

Quando uma referência é feita a uma variável TLS dentro de um thread, a referência acessa a cópia da variável associada a esse thread. Ele não acessará ou atualizará uma cópia da variável associada a nenhum outro thread.

Como cada variável TLS está associada a um thread, a sincronização (conforme descrito no Capítulo 14, "Sincronização de Armazenamento Compartilhado", na página 151) geralmente não é uma preocupação. A sincronização pode se tornar necessária, no entanto, se o endereço de uma variável TLS for passado para outro thread.

Capítulo 8. Gerenciamento de armazenamento

**104 Conceitos** ILE IBM i 7.1

# Capítulo 9. Gerenciamento de exceções e condições

Este capítulo fornece detalhes adicionais sobre o tratamento de exceções e o tratamento de condições. Antes de ler este capítulo, leia os conceitos avançados descritos em "Tratamento de Erros" na página 34.

A arquitetura de mensagem de exceção do sistema operacional é usada para implementar o tratamento de exceções e o tratamento de condições. Há casos em que o tratamento de exceções e o tratamento de condições interagem. Por exemplo, um manipulador de condição ILE registrado com a API vinculável Registrar um manipulador de condição escrito pelo usuário (CEEHDLR) é usado para manipular uma mensagem de exceção enviada com o botão Enviar API de Mensagem de Programa (QMHSNDPM). Essas interações são explicadas neste capítulo. O termo manipulador de exceção é usado neste capítulo para significar um  *manipulador* de exceção do sistema operacional ou um manipulador de condição ILE.

# Manipular cursores e cursores de retomada

Para processar exceções, o sistema usa dois ponteiros chamados cursor do identificador e cursor de retomada. Esses ponteiros acompanham o progresso do tratamento de exceções. Você precisa entender o uso do cursor do identificador e do cursor de retomada em determinados cenários avançados de tratamento de erros. Esses conceitos são usados para explicar recursos adicionais de tratamento de erros em tópicos posteriores.

O  **cursor do identificador** é um ponteiro que mantém o controle do manipulador de exceções atual. À medida que o sistema procura um manipulador de exceções disponível, ele move o cursor do identificador para o próximo manipulador na lista de manipuladores de exceção definida por cada entrada de pilha de chamadas. Esta lista pode conter:

v Manipuladores de monitor direto v Manipuladores de condição ILE v Manipuladores específicos de HLL

O cursor do identificador se move para baixo na lista de manipuladores de exceção para manipuladores de prioridade mais baixa até que a exceção seja tratada. Se a exceção não for manipulada por nenhum dos manipuladores de exceção que foram definidos para uma entrada de pilha de chamadas, o cursor do identificador será movido para o primeiro (mais alto priority) manipulador para a entrada de pilha de chamadas anterior.

O cursor de retomada é um ponteiro que mantém o controle do local atual no qual o manipulador de exceções pode retomar o processamento depois de manipular a exceção.  Normalmente, o sistema define o cursor de retomada para a próxima instrução após a ocorrência de uma exceção . Para entradas de pilha de chamadas acima do procedimento que incorreu na exceção, o ponto de retomada é diretamente após o procedimento ou chamada de programa que atualmente suspendeu o procedimento ou programa. Para mover o cursor de retomada para um ponto de retomada anterior, use a API vinculável Mover Cursor de Retomada (CEEMRCR ).

A Figura 44 na página 106 mostra um exemplo do cursor da alça e do cursor de retomada.

© Direitos autorais IBM Corp. 1997, 2010

.

**Pilha de chamadas |**

Procedimento P1

CALLPRC P2

Procedimento P2

CALLPRC P3

Exceção

Treinador

Lista

Manéjar

Cursor

Retomar

Cursor

Procedimento P3

Ocorreu uma exceção

ILE

ILE

ILE

*Ponto de Currículo*

*Ponto de Currículo*

*Ponto de Currículo*

.

.

Manipulador de exceções

Processo P10

RV2W1044-0

*Figura 44. Exemplo* de  *cursor de manipulação e cursor de retomada*

O cursor do identificador está atualmente no segundo manipulador de exceções definido na lista de prioridades do manipulador de exceções para o procedimento P2. O procedimento do manipulador P10 é atualmente chamado pelo sistema. Se o procedimento P10 manipular a exceção e retornar, o controle irá para o local atual do cursor de retomada definido no procedimento P3. Este exemplo pressupõe que o procedimento P3 percolou a exceção ao procedimento P2.

O procedimento do manipulador de exceções P10 pode modificar o cursor de retomada com a API vinculável Mover Cursor de Retomada (CEEMRCR ). Duas opções são fornecidas com esta API. Um manipulador de exceções pode modificar o cursor de retomada para uma das seguintes opções:

v A entrada da pilha de chamadas que contém o cursor do identificador v A entrada da pilha de chamadas antes do cursor do identificador

Na Figura 44, você pode modificar o cursor de retomada para o procedimento P2 ou P1. Depois que o cursor de retomada é modificado e a exceção é marcada como manipulada, um retorno normal do manipulador de exceções retorna o controle para o novo ponto de retomada.

# Ações do manipulador de exceções

Quando o manipulador de exceção é chamado pelo sistema, você pode executar várias ações para manipular a exceção. Por exemplo, as extensões ILE C oferecem suporte a ações de controle, manipuladores de ponto de ramificação e monitoramento por ID de mensagem. As possíveis ações descritas aqui pertencem a qualquer um dos seguintes tipos de manipuladores:

v Manipulador de monitor direto v Manipulador de condição ILE v manipulador específico de HLL

## Como retomar o processamento

Se você determinar que o processamento pode continuar, poderá retomar no local atual do cursor de retomada.

Antes de retomar o processamento, a mensagem de exceção deve ser alterada para indicar que ela foi

manuseado. Certos tipos de manipuladores exigem que você altere explicitamente a mensagem de exceção para indicar que a mensagem foi manipulada. Para outros tipos de manipulador, o sistema pode alterar a mensagem de exceção antes que o manipulador seja chamado.

Para um manipulador de monitor direto, você pode especificar uma ação a ser executada para a mensagem de exceção. Essa ação pode ser chamar o manipulador, manipular a exceção antes de chamar o manipulador ou manipular a exceção e retomar o programa. Se a ação for apenas para chamar o manipulador, você ainda poderá manipular a exceção usando a API Change Exception Message (QMHCHGEM) ou a API vinculável CEE4HC (Handle Condição). Você pode alterar o ponto de retomada em um manipulador de monitor direto usando a API vinculável Mover Cursor de Retomada (CEEMRCR ). Depois de fazer essas alterações, você continua o processamento retornando do manipulador de exceções.

Para um manipulador de condição ILE, você continua o processamento definindo um valor de código de retorno e retornando ao sistema. Para obter informações sobre valores reais de código de retorno para a API vinculável Registrar um manipulador de condição escrito pelo usuário (CEEHDLR), consulte a coleção de tópicos da API na categoria Programação do Centro de Informações do IBM i.

Para um manipulador específico de HLL, a mensagem de exceção é alterada para indicar que ele foi manipulado antes de seu manipulador ser chamado. Para determinar se você pode modificar o cursor de retomada de um manipulador específico de HLL, consulte o guia do programador ILE HLL .

## Como Realizar uma Mensagem

Se você determinar que uma mensagem de exceção não é reconhecida pelo manipulador, poderá incluir a mensagem de exceção no próximo manipulador disponível. Para que a percolação ocorra, a mensagem de exceção não deve ser considerada como uma mensagem manipulada. Outros manipuladores de exceção na mesma ou em entradas de pilha de chamadas anteriores têm a chance de manipular a mensagem de exceção. A técnica para percolar uma mensagem de exceção varia dependendo do tipo de manipulador de exceções.

Para um manipulador de monitor direto, não altere a mensagem de exceção para indicar que ela foi manipulada. Um retorno normal do manipulador de exceções faz com que o sistema percolate a mensagem. A mensagem é percolada para o próximo manipulador de exceções na lista de manipuladores de exceção para sua entrada de pilha de chamadas. Se o manipulador estiver no final da lista de manipuladores de exceções, a mensagem será percolada para o primeiro manipulador de exceções na entrada de pilha de chamadas anterior.

Para um manipulador de condição ILE, você comunica uma ação de percolação definindo um valor de código de retorno e retornando ao sistema. Para obter informações sobre os valores reais de código de retorno para a API vinculável Registrar um manipulador de condição escrito pelo usuário (CEEHDLR), consulte a coleção de tópicos da API na categoria Programação de o Centro de Informações do IBM i.

Para um manipulador específico de HLL, talvez não seja possível visualizar uma mensagem de exceção. Se você pode percolar uma mensagem depende se o HLL marca a mensagem como manipulada antes de seu manipulador ser chamado. Se você não declarar um manipulador específico de HLL, seu HLL poderá incluir a mensagem de exceção não tratada. Consulte o manual de referência do ILE HLL para determinar as mensagens de exceção que seu manipulador específico do HLL pode manipular.

## Como promover uma mensagem

Em determinadas situações limitadas, você pode optar por modificar a mensagem de exceção para uma mensagem diferente. Essa ação marca a mensagem de exceção original como manipulada e reinicia o processamento de exceção com uma nova mensagem de exceção. Essa ação é permitida somente de manipuladores de monitor diretos e manipuladores de condição ILE.

Para manipuladores de monitor diretos, use a API Promote Message (QMHPRMM) para promover uma mensagem. O sistema pode promover apenas tipos de mensagem de status e escape. Com essa API, você tem algum controle sobre o posicionamento do cursor do identificador que é usado para continuar o processamento de exceções. Consulte o tópico API na categoria Programação do Centro de Informações do IBM i.

Para um manipulador de condição ILE, você comunica a ação de promoção definindo um valor de código de retorno e retornando ao sistema. Para obter informações sobre os valores reais de código de retorno para a API vinculável Registrar um manipulador de condição escrito pelo usuário (CEEHDLR), consulte a coleção de tópicos da API na categoria Programação de o Centro de Informações do IBM i.

# Ações padrão para exceções não tratadas

Se uma mensagem de exceção for percolada ao limite de controle, o sistema executará uma ação padrão. Se a exceção for uma mensagem de notificação, o sistema enviará a resposta padrão , manipulará a exceção e permitirá que o remetente da mensagem de notificação continue o processamento . Se a exceção for uma mensagem de status, o sistema manipulará a exceção e permitirá que o remetente da mensagem de status continue o processamento. Se a exceção for uma mensagem de escape, o sistema manipula a mensagem de escape e envia uma mensagem de verificação de função de volta para onde o cursor de retomada está posicionado no momento . Se a exceção não tratada for uma verificação de função, todas as entradas na pilha até o limite de controle serão canceladas e a mensagem de escape CEE9901 será enviado para a próxima entrada de pilha anterior.

A Tabela 10 contém as respostas padrão que o sistema toma quando uma exceção não é tratada em um limite de controle.

*Tabela 10. Respostas padrão a exceções não tratadas*

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo** de mensagem | **Gravidade da condição** | **Condição levantada pelo**  **Sinalizar uma condição**  **(CEESGL) API** **vinculável** | **Exceção originada de qualquer outra fonte** |
| Estado | 0 (Mensagem informativa) | Retorne a condição não tratada. | Retomar sem registrar a mensagem. |
| Estado | 1 (Atenção) | Retorne a condição não tratada. | Retomar sem registrar a mensagem. |
| Notificar | 0 (Mensagem informativa) | Não aplicável. | Registre a mensagem de notificação e envie a resposta padrão. |
| Notificar | 1 (Atenção) | Não aplicável. | Registre a mensagem de notificação e envie a resposta padrão. |
| Escapar | 2 (Erro) | Retorne a condição não tratada. | Registre a mensagem de escape e envie uma mensagem de verificação de função para a entrada da pilha de chamadas do ponto de retomada atual. |
| Escapar | 3 (Erro grave) | Retorne a condição não tratada. | Registre a mensagem de escape e envie uma mensagem de verificação de função para a entrada da pilha de chamadas do ponto de retomada atual. |
| Escapar | 4 (Erro crítico de ILE) | Registrar a mensagem de escape e Registrar a mensagem de escape e enviar uma verificação de função enviar uma mensagem de verificação de função para a mensagem de pilha de chamadas para a pilha de chamadas  entrada da entrada do currículo atual do ponto de currículo atual . ponto. | |
| Verificação de função | 4 (Erro crítico de ILE) | Não aplicável Finalizar o aplicativo, e  enviar a mensagem CEE9901 para o chamador do limite de controle. | |

**Nota:** Quando o aplicativo é encerrado por uma verificação de função não tratada, o grupo de ativação é excluído se o limite de controle for a entrada de pilha de chamadas mais antiga na ativação grupo.

# Exceções aninhadas

Uma exceção **aninhada**  é uma exceção que ocorre enquanto outra exceção está sendo tratada. Quando isso acontece, o processamento da primeira exceção é temporariamente suspenso. O sistema salva todas as informações associadas, como os locais do cursor da alça e do cursor de retomada. O tratamento de exceções começa novamente com a exceção gerada mais recentemente. Novos locais para o cursor da alça e o cursor de retomada são definidos pelo sistema. Uma vez que a nova exceção tenha sido tratada corretamente, as atividades de manipulação para a exceção original normalmente são retomadas.

Quando ocorre uma exceção aninhada, ambos os itens a seguir ainda estão na pilha de chamadas: v A entrada de pilha de chamadas associada à exceção original v A entrada de pilha de chamadas associada à manipulador de exceções original

Para reduzir a possibilidade de loops de manipulação de exceções , o sistema interrompe a percolação de uma exceção aninhada na entrada de pilha de chamadas do manipulador de exceções original. Em seguida, o sistema promove a exceção aninhada para uma mensagem de verificação de função e percola a mensagem de verificação de função para a mesma entrada de pilha de chamadas. Se você não manipular a exceção aninhada ou a mensagem de verificação de função, o sistema encerrará o aplicativo chamando a API vinculável Abnormal End (CEE4ABN ). Nesse caso, a mensagem CEE9901 é enviada ao chamador do limite de controle.

Se você mover o cursor de retomada durante o processamento da exceção aninhada, poderá modificar implicitamente a exceção original. Para fazer com que isso ocorra, faça o seguinte:

1. Mover o cursor de retomada para uma entrada de pilha de chamadas anterior à entrada de pilha de chamadas que incorreu na exceção original
2. Retomar o processamento retornando do manipulador

# Manipulação de Condição

*As condições* ILE são mensagens de exceção do sistema operacional representadas de maneira independente do sistema. Um token de condição ILE é usado para representar uma condição ILE. *O* tratamento de condições refere-se às funções ILE que você pode usar para manipular erros separadamente do tratamento de erros específicos do idioma. Outros sistemas implementaram essas funções. Você pode usar o tratamento de condições para aumentar a portabilidade de seus aplicativos entre sistemas que implementaram o tratamento de condições.

O tratamento de condição ILE inclui as seguintes funções: v Capacidade de registrar dinamicamente um manipulador de condição ILE v Capacidade de sinalizar uma condição ILE v Arquitetura de token de condição

v Códigos de feedback de token de condição opcionais para APIs ILE vinculáveis

Essas funções são descritas nos tópicos a seguir.

## Como as condições são representadas

O  **token de** condição ILE é um tipo de dados compostos de 12 bytes que contém campos estruturados para transmitir aspectos de uma condição. Tais aspectos podem ser sua gravidade, seu número de mensagem associado e informações específicas para a instância dada da condição. O token de condição é usado para comunicar essas informações sobre uma condição ao sistema, aos serviços de mensagens, às APIs vinculáveis e aos procedimentos. As informações retornadas no parâmetro fc opcional de todas as APIs vinculáveis ILE, por exemplo, são comunicadas usando um token de condição.

Se uma exceção for detectada pelo sistema operacional ou pelo hardware, um token de condição correspondente será criado automaticamente pelo sistema. Você também pode criar um token de condição usando a API vinculável Construct a Condition Token (CEENCOD). Em seguida, você pode sinalizar uma condição para o sistema retornando o token por meio da API vinculável Signal a Condition (CEESGL ).

### Layout de um token de condição

A Figura 45 exibe um mapa do token de condição. A posição do bit inicial é mostrada para cada campo.

0

0

32

3437

40

64

16

Condition\_ID

MsgSev

Msg\_No

I\_S\_Info

Facility\_ID

A ID da condição ILE

sempre tem o formato do caso 1:

Controle

Severidade

Caso

RV2W1032-2

*Figura 45. Layout de token de* *condição* ILE

Cada token de condição contém os componentes indicados na Figura 45:

##### Condition\_ID

Um identificador de 4 bytes que, com o Facility\_ID, descreve a condição que o token se comunica. As APIs vinculáveis ao ILE e a maioria dos aplicativos produzem condições de caso 1.

**Caso Um** campo de 2 bits que define o formato da parte Condition\_ID do token. As condições de ILE são sempre o caso 1.

##### Severidade

Um inteiro binário de 3 bits que indica a gravidade da condição. Os campos Gravidade e MsgSev contêm as mesmas informações. Consulte a Tabela 10 na página 108 para obter uma lista de severidades de condição de ILE. Consulte a Tabela 12 na página 111 e a Tabela 13 na página 111 para as severidades de mensagem do sistema operacional correspondentes.

##### Controle

Um campo de 3 bits que contém sinalizadores que descrevem ou controlam vários aspectos do tratamento de condições. O terceiro bit especifica se o Facility\_ID foi atribuído pela IBM.

##### Facility\_ID

Uma cadeia alfanumérica de 3 caracteres que identifica o recurso que gerou a condição. O Facility\_ID indica se a mensagem foi gerada pelo sistema ou por um tempo de execução HLL. A Tabela 11 da página 111 lista os IDs de instalação usados no ILE.

##### I\_S\_Info

Um campo de 4 bytes que identifica as informações específicas da instância associadas a uma determinada instância da condição. Este campo contém a chave de referência para a instância da mensagem associada ao token de condição. Se a chave de referência da mensagem for zero, não haverá nenhuma mensagem associada.

##### MsgSev

Um inteiro binário de 2 bytes que indica a gravidade da condição. MsgSev e Severity contêm as mesmas informações. Consulte a Tabela 10 na página 108 para obter uma lista de severidades de condição de ILE. Consulte a Tabela 12 na página 111 e a Tabela 13 na página 111 para as severidades de mensagem do sistema operacional correspondentes.

##### Msg\_No

Um número binário de 2 bytes que identifica a mensagem associada à condição. A combinação de Facility\_ID e Msg\_No identifica exclusivamente uma condição.

A Tabela 11 contém as IDs de recurso usadas nos tokens de condição ILE e no prefixo das mensagens.

*Tabela 11.*   *IDs* de  *recurso usados em mensagens* e  *tokens* *de condição ILE*

|  |  |
| --- | --- |
| **ID** da instalação | **Instalação** |
| CEE (CEE) | Biblioteca comum ILE |
| CPF | Mensagem do IBM i |
| MCH | Mensagem de exceção de máquina do IBM i |

## Teste de token de condição

Você pode testar um token de condição retornado de uma API vinculável para o seguinte:

#### Êxito

Para testar o sucesso, determine se os primeiros 4 bytes são zero. Se os primeiros 4 bytes forem zero, o restante do token de condição será zero, indicando que uma chamada bem-sucedida foi feita para a API vinculável.

#### Tokens equivalentes

Para determinar se dois tokens de condição são equivalentes (ou seja, o mesmo *tipo* de token de condição, mas não a mesma *instância* do token de condição), compare os primeiros 8 bytes de cada token de condição uns com os outros. Esses bytes são os mesmos para todas as instâncias de uma determinada condição.

#### Tokens Iguais

Para determinar se dois tokens de condição são iguais (ou seja, eles representam a mesma instância de uma condição), compare todos os 12 bytes de cada token de condição entre si. Os últimos 4 bytes podem mudar de instância para instância de uma condição.

## Relação das condições ILE com as mensagens do sistema operacional

Uma mensagem é associada a todas as condições geradas no ILE. O token de condição contém uma ID exclusiva que o ILE usa para gravar uma mensagem associada à condição no arquivo de mensagem.

O formato de cada mensagem de tempo de execução é **FFFxxxx**:

**FFF** O ID do recurso, um ID de 3 caracteres que é usado por todas as mensagens geradas nos idiomas ILE e ILE. Consulte a Tabela 11 para obter uma lista de IDs e recursos correspondentes.

**xxxx** O número da mensagem de erro. Este é um número hexadecimal que identifica a mensagem de erro associada à condição.

A Tabela 12 e a Tabela 13 mostram como a gravidade da condição de ILE é mapeada para a gravidade da mensagem.

*Tabela 12. Mapeando*  as gravidades *da mensagem* \*ESCAPE  *para as severidades da condição ILE*

|  |  |  |
| --- | --- | --- |
| **Da Gravidade** **da Mensagem do IBM i** | **Para a gravidade** da **condição ILE** | **Para a Gravidade** **da Mensagem do IBM i** |
| 0-29 | 2 | 20 |
| 30-39 | 3 | 30 |
| 40-99 | 4 | 40 |

*Tabela 13. Mapeando \*STATUS e \*NOTIFY Gravidades de Mensagem para Severidades* *de Condição ILE*

|  |  |  |
| --- | --- | --- |
| **Da Gravidade** **da Mensagem do IBM i** | **Para a gravidade** da **condição ILE** | **Para a Gravidade** **da Mensagem do IBM i** |
| 0 | 0 | 0 |
| 1-99 | 1 | 10 |

## Mensagens do IBM i e o Código de Feedback da API Bindable

Como entrada para uma API vinculável, você tem a opção de codificar um código de feedback e usar o código de feedback como uma verificação de código de retorno (ou feedback) em um procedimento. O código de feedback é um valor de token de condição que é fornecido para flexibilidade na verificação de retornos de chamadas para outros procedimentos. Em seguida, você pode usar o código de feedback como entrada para um token de condição. Se o código de feedback for omitido na chamada para uma API vinculável e ocorrer uma condição, uma mensagem de exceção será enviada ao chamador da API vinculável.

Se você codificar o parâmetro de código de feedback em seu aplicativo para receber informações de feedback de uma API vinculável, a seguinte sequência de eventos ocorrerá quando uma condição for gerada :

1. Uma mensagem informativa é enviada ao chamador da API, comunicando a mensagem associada à condição.
2. A API vinculável na qual a condição ocorreu cria um token de condição para a condição. A API vinculável coloca as informações na área de informações específicas da instância. As informações específicas da instância do token de condição são a chave de referência da mensagem informativa. Isso é usado pelo sistema para reagir à condição.
3. Se uma condição detectada for crítica (a gravidade é 4), o sistema enviará uma mensagem de exceção ao chamador da API vinculável.
4. Se uma condição detectada não for crítica (gravidade menor que 4), o token de condição será retornado à rotina chamada API vinculável.
5. Quando o token de condição é retornado ao seu aplicativo, você tem as seguintes opções:

v Ignore-o e continue o processamento.

v Sinalize a condição usando a API vinculável Signal a Condition (CEESGL ).

v Obtenha, formate e despache a mensagem para exibição usando a API vinculável Get, Format, and Dispatch a Message (CEEMSG ).

v Armazene a mensagem em uma área de armazenamento usando a API vinculável Obter uma Mensagem (CEEMGET ).

v Use a API vinculável Dispatch a Message (CEEMOUT) para despachar uma mensagem definida pelo usuário para um destino que você especificar.

v Quando o chamador da API recupera o controle, a mensagem informativa é removida e não aparece no log de trabalho.

Se você omitir o parâmetro de código de feedback quando estiver chamando uma API vinculável , a API vinculável enviará uma mensagem de exceção ao chamador da API vinculável.

# Capítulo 10. Considerações sobre depuração

O depurador de origem é usado para depurar OPM, ILE e programas de serviço. Os comandos CL ainda podem ser usados para depurar programas OPM (modelo de programa original).

Este capítulo apresenta várias considerações sobre o depurador de origem. Informações sobre como usar o depurador de origem podem ser encontradas nas informações on-line e no guia do programador para a linguagem de alto nível (HLL) ILE que você está usando. Informações sobre os comandos a serem usados para uma tarefa específica (por exemplo, criar um módulo) podem ser encontradas no guia do programador ILE HLL.

# Modo de depuração

Para usar o depurador de origem, sua sessão deve estar no modo de depuração. **O modo** de depuração é um ambiente especial no qual as funções de depuração do programa podem ser usadas além das funções normais do sistema.

Sua sessão é colocada no modo de depuração quando você executa o comando Iniciar depuração (STRDBG).

## Ambiente de depuração

Um programa pode ser depurado em um dos dois ambientes:

v O ambiente de depuração do OPM. Todos os programas OPM são depurados neste ambiente, a menos que os programas OPM sejam explicitamente adicionados ao ambiente de depuração ILE.

v O ambiente de depuração ILE. Todos os programas ILE são depurados neste ambiente. Além disso, um programa OPM é depurado neste ambiente se todos os seguintes critérios forem atendidos:

* É um programa CL, COBOL ou RPG.
* Ele é compilado com dados de depuração de origem do OPM.
* O parâmetro OPMSRC (OPM source level debug) do comando STRDBG é definido como \*YES.

O ambiente de depuração ILE fornece suporte à depuração no nível de origem. O recurso de depuração vem diretamente de exibições de instrução, origem ou lista do código.

Uma vez que um programa OPM esteja no ambiente de depuração ILE, o sistema fornecerá depuração perfeita dos programas ILE e OPM por meio da mesma interface de usuário. Para obter informações sobre como usar o depurador de origem para programas OPM no ambiente de depuração ILE, consulte a ajuda on-line ou o guia do programador para a linguagem de alto nível ILE equivalente (HLL) que você está usando para a linguagem OPM: CL, COBOL ou RPG.

## Adição de programas ao modo de depuração

Um programa deve ser adicionado ao modo de depuração antes de poder ser depurado. Programas OPM, programas ILE e programas de serviço ILE podem estar no modo de depuração ao mesmo tempo. Até 20 programas OPM podem estar no modo de depuração de uma só vez no ambiente de depuração do OPM. O número de programas ILE, programas de serviço e programas OPM no ambiente de depuração ILE que podem estar no modo de depuração de uma só vez não é limitado. No entanto, a quantidade máxima de dados de depuração que é suportada de uma só vez é de 16MB por módulo.

Você deve ter autoridade \*CHANGE para um programa ou programa de serviço para adicioná-lo ao modo de depuração. Um programa ou programa de serviço pode ser adicionado ao modo de depuração quando é interrompido na pilha de chamadas.

Os programas ILE e os programas de serviço são acessados pelo depurador de origem, um módulo de cada vez. Quando você estiver depurando um programa ILE ou programa de serviço, talvez seja necessário depurar um módulo em outro programa ou programa de serviço. Esse segundo programa ou programa de serviço deve ser adicionado ao modo de depuração antes que o módulo no segundo programa possa ser depurado.

© Direitos autorais IBM Corp. 1997, 2010

Quando o modo de depuração termina, todos os programas são removidos do modo de depuração.

# Como a observabilidade e a otimização afetam a depuração

O nível de otimização e a observabilidade de dados de depuração de módulos acoplados afetam sua capacidade de depurar um programa.

## Níveis de otimização

Com níveis mais altos de otimização, você não pode alterar variáveis e talvez não consiga exibir o valor real de uma variável durante a depuração . Quando você estiver depurando, defina o nível de otimização como 10 (\*NENHUM). Isso fornece o nível mais baixo de desempenho para os procedimentos no módulo, mas permite que você exiba e altere as variáveis com precisão. Depois de concluir a depuração, defina o nível de otimização como 30 (\*FULL) ou 40. Isso fornece o mais alto nível de desempenho para os procedimentos no módulo.

## Depurar a criação e remoção de dados

Os dados de depuração são armazenados com cada módulo e são gerados quando um módulo é criado. Para depurar um procedimento em um módulo que foi criado sem dados de depuração, você deve recriar o módulo com dados de depuração e, em seguida, revincular o módulo ao programa ILE ou programa de serviço. Não é necessário recompilar todos os outros módulos no programa ou programa de serviço que já tenham dados de depuração.

Para remover dados de depuração de um módulo, recrie o módulo sem dados de depuração ou use o comando Change Module (CHGMOD).

## Visualizações do módulo

Os níveis de dados de depuração disponíveis podem variar para cada módulo em um programa ILE ou programa de serviço. Os módulos são compilados separadamente e podem ser produzidos com diferentes compiladores e opções. Esses níveis de dados de depuração determinam quais exibições são produzidas pelo compilador e quais **exibições** são exibidas pelo depurador de origem. Os valores possíveis são:

**\*NENHUM**

Nenhuma exibição de depuração é produzida.

#### \*STMT

Nenhuma fonte é exibida pelo depurador, mas os pontos de interrupção podem ser adicionados usando nomes de procedimento e números de instrução encontrados na listagem do compilador. A quantidade de dados de depuração armazenados com esse modo de exibição é a quantidade mínima de dados necessária para a depuração.

#### \*FONTE

O depurador de código-fonte exibirá o código-fonte se os arquivos de código-fonte usados para compilar o módulo ainda estiverem presentes no sistema.

**\*LIST** A visualização de lista é produzida e armazenada com o módulo. Isso permite que o depurador de origem exiba a origem mesmo que os arquivos de origem usados para criar o módulo não estejam presentes no sistema. Essa exibição pode ser útil como uma cópia de backup se o programa for alterado. No entanto, a quantidade de dados de depuração pode ser bastante grande, especialmente se outros arquivos forem expandidos para a listagem. As opções do compilador usadas quando os módulos foram criados determinam se as inclusões são expandidas. Os arquivos que podem ser expandidos incluem arquivos DDS e incluem arquivos (como ILE C inclui, arquivos ILE RPG /COPY e arquivos ILE COBOL COPY).

**\*TODAS** as exibições de depuração são produzidas. Quanto ao modo de exibição de lista, a quantidade de dados de depuração pode ser muito grande.

O ILE RPG também tem uma opção de depuração \*COPY que produz uma exibição de código-fonte e uma exibição de cópia. O modo de exibição de cópia é um modo de exibição de depuração que tem todos os membros de origem /COPY incluídos.

## Depuração entre trabalhos

Talvez você queira usar um trabalho separado para depurar programas em execução em seu trabalho ou em lotes. Isso é muito útil quando você deseja observar a função de um programa sem a interferência de painéis de depurador. Por exemplo, os painéis ou janelas que um aplicativo exibe podem sobrepor ou ser sobrepostos pelos painéis do depurador durante a etapa ou em pontos de interrupção. Você pode evitar esse problema iniciando um trabalho de serviço e iniciando o depurador em um trabalho diferente daquele que está sendo depurado. Para obter informações sobre isso, consulte o apêndice sobre testes no livro CL Programming.

## Suporte a depuradores OPM e ILE

O suporte ao depurador OPM e ILE permite a depuração no nível de origem dos programas OPM por meio das APIs do depurador ILE. Para obter informações sobre APIs do Depurador ILE, consulte o tópico API na categoria Programação do Centro de Informações do IBM i. O suporte ao depurador OPM e ILE fornece depuração perfeita dos programas ILE e OPM por meio da mesma interface do usuário. Para usar esse suporte, você deve compilar um programa OPM com o compilador RPG, COBOL ou CL. Você deve definir o parâmetro OPTION como \*SRCDBG ou \*LSTDBG para a compilação.

## Suporte do Watch

O suporte a Inspeção fornece a capacidade de interromper a execução do programa quando o conteúdo de um local de armazenamento especificado é alterado. O local de armazenamento é especificado pelo nome de uma variável de programa. A variável de programa é resolvida para um local de armazenamento e o conteúdo nesse local é monitorado quanto a alterações. Se o conteúdo no local de armazenamento for alterado, a execução será interrompida. A origem do programa interrompida é exibida no ponto de interrupção e a linha de origem realçada será executada após a instrução que alterou o local de armazenamento .

# Exceções não monitoradas

Quando ocorre uma exceção não monitorada, o programa que está sendo executado emite uma verificação de função e envia uma mensagem para o log de trabalho. Se você estiver no modo de depuração e os módulos do programa foram criados com dados de depuração, o depurador de origem mostrará a exibição Origem do módulo de exibição. O programa é adicionado ao modo de depuração, se necessário. O módulo apropriado é mostrado no visor com a linha afetada realçada. Em seguida, você pode depurar o programa.

# Restrição de globalização para depuração

Se existir uma das seguintes condições:

v O identificador de conjunto de caracteres codificado (CCSID) do trabalho de depuração é 290, 930 ou 5026 (Japan Katakana) v A página de código da descrição do dispositivo usada para depuração é 290, 930 ou 5026 (Japão Katakana) comandos de depuração, funções e literais hexadecimais devem ser inseridos em maiúsculas. Por exemplo:

INTERVALO 16 QUANDO var=X'A1B2'

EVAL var: X

A restrição acima para páginas de código do Japão Katakana não se aplica ao usar nomes de identificadores em comandos de depuração (por exemplo, EVAL). No entanto, ao depurar módulos ILE RPG, ILE COBOL ou ILE CL, os nomes de identificador nos comandos de depuração são convertidos em maiúsculas pelo depurador de origem e, portanto, podem ser reexibidos de forma diferente.

Capítulo 10. Considerações sobre depuração

**116 Conceitos** de ILE IBM i 7.1

# Capítulo 11. Escopo de gerenciamento de dados

Este capítulo contém informações sobre os recursos de gerenciamento de dados que podem ser usados por um programa ILE ou programa de serviço. Antes de ler este capítulo, você deve entender os conceitos de escopo de gerenciamento de dados descritos em "Regras de escopo de gerenciamento de dados" na página 41.

Os detalhes de cada tipo de recurso são deixados para o guia de cada programador ILE HLL.

# Recursos comuns de gerenciamento de dados

Este tópico identifica todos os recursos de gerenciamento de dados que seguem as regras de escopo de gerenciamento de dados. A seguir a cada recurso há uma breve descrição de como especificar o escopo. Detalhes adicionais para cada recurso podem ser encontrados nas publicações referidas.

#### Abrir operações de arquivo

| As operações de arquivo aberto resultam na criação de um recurso temporário chamado de dados abertos

| caminho (ODP). Você pode iniciar a função open usando verbos abertos HLL, o arquivo Open Query

| (OPNQRYF) ou o comando Open Data Base File (OPNDBF). O ODP tem como escopo

| o grupo de ativação do programa que abriu o arquivo. Para programas OPM ou ILE que são executados em um

| grupo de ativação padrão, o ODP tem o escopo definido para o número de nível de chamada. Para alterar o escopo de

| HLL verbos abertos, você pode usar uma substituição. Você pode especificar o escopo usando o escopo aberto | (OPNSCOPE) parâmetro em todos os comandos de substituição, o comando OPNDBF e o OPNQRYF

| comando.

#### Substitui

| As substituições têm o escopo definido para o nível de chamada, o nível do grupo de ativação ou o nível do trabalho. Para especificar

| substituir o escopo, use o parâmetro de escopo de substituição (OVRSCOPE) em qualquer comando de substituição. Se

| você não especifica o escopo explícito, o escopo da substituição depende de onde o sistema emite

| a substituição. Se o sistema emitir a substituição de um grupo de ativação padrão, o escopo será definido para o | nível de chamada. Se o sistema emitir a substituição de qualquer outro grupo de ativação, o escopo será definido de acordo com o | nível do grupo de ativação.

#### Definições de compromisso

As definições de compromisso dão suporte ao escopo para o nível do grupo de ativação e ao escopo para o nível do trabalho. O nível de escopo é especificado com o parâmetro de escopo de controle (CTLSCOPE) no comando Start Commitment Control (STRCMTCTL). Para obter mais informações sobre definições de compromisso, consulte o tópico Backup e Recuperação no Centro de Informações do IBM i.

#### Cursores SQL locais

| Os cursores SQL podem ser declarados e usados em aplicativos criados pelo usuário ou no Database | programas ILE C construídos ou programas de serviço ILE C criados em nome do CREATE PROCEDURE, | CREATE FUNCTION ou instruções CREATE TRIGGER LANGUAGE SQL.

| Para todos os tipos de aplicativos SQL, há controles de escopo de cursor SQL. O cursor controla | incluir uma opção para definir o escopo dos cursores para o grupo de ativação, bem como várias outras opções de escopo.

| Para obter detalhes completos, consulte IBM i DB2® for i SQL Reference.

#### Conexões SQL remotas

As conexões remotas usadas com cursores SQL têm o escopo de um grupo de ativação implicitamente como parte do processamento SQL normal. Isso permite que existam várias conversas entre um trabalho de origem e vários trabalhos de destino ou vários sistemas.

#### Gerenciador de interface do usuário

As APIs do Open Print Application (QUIOPNPA) e do Open Display Application oferecem suporte a um parâmetro de escopo do aplicativo. Essas APIs podem ser usadas para definir o escopo do UIM (gerenciador de interface do usuário)

© Direitos autorais IBM Corp. 1997, 2010

aplicativo para um grupo de ativação ou para o trabalho. Para obter mais informações sobre o gerenciador de interface do usuário, consulte o tópico API na categoria Programação do Centro de Informações do IBM i.

#### Abrir links de dados (abrir gerenciamento de arquivos)

| A API Enable Link (QOLELINK) habilita um link de dados. Se você usar essa API de dentro de um

| grupo de ativação não padrão, o link de dados tem o escopo para esse grupo de ativação. Se você usar esta API

| de dentro de um grupo de ativação padrão, o link de dados tem o escopo definido para o nível de chamada. Para mais | informações sobre links de dados abertos, consulte o tópico API na categoria Programação do IBM | i Centro de Informação.

#### Conversas de comunicação da Common Programming Interface (CPI)

O grupo de ativação que inicia uma conversa é o proprietário dessa conversa. O grupo de ativação que habilita um link por meio da API Enable Link (QOLELINK) possui o link. Para obter informações sobre conversas de Comunicações CPI (Common Programming Interface), consulte a coleção de tópicos da API na categoria Programação do Centro de Informações do IBM i.

#### Sistema de arquivos hierárquico

A API Open Stream File (OHFOPNSF) gerencia arquivos do sistema de arquivos hierárquico (HFS). Você pode usar o parâmetro open information (OPENINFO) nessa API para controlar o escopo para o grupo de ativação ou o nível de trabalho. Para obter mais informações sobre o sistema de arquivos hierárquico, consulte o tópico API na categoria Programação do Centro de Informações do IBM i.

# Escopo de Controle de Compromisso

O ILE introduz duas mudanças para o controle de compromisso:

v Múltiplas definições de compromisso independentes por trabalho. As transações podem ser confirmadas e revertidas independentemente umas das outras. Antes do ILE, apenas uma única definição de compromisso era permitida por trabalho.

v Se as alterações estiverem pendentes quando um grupo de ativação terminar normalmente, o sistema confirmará implicitamente as alterações. Antes do ILE, o sistema não confirmava as alterações.

O controle de compromisso permite que você defina e processe alterações em recursos, como arquivos de banco de dados ou tabelas, como uma única transação . Uma **transação** é um grupo de alterações individuais em objetos no sistema que devem aparecer para o usuário como uma única alteração atômica. O controle de compromisso garante que uma das seguintes situações ocorra no sistema:

v O grupo inteiro de alterações individuais ocorre (uma operação **de confirmação**) v Nenhuma das alterações individuais ocorre (uma **operação de reversão**)

Vários recursos podem ser alterados sob controle de compromisso usando programas OPM e programas ILE.

O comando Start Commitment Control (STRCMTCTL) possibilita que os programas executados em um trabalho façam alterações sob controle de compromisso. Quando o controle de compromisso é iniciado usando o comando STRCMTCTL, o sistema cria uma  **definição** **de** compromisso**.**  Cada definição de compromisso é conhecida apenas pelo trabalho que emitiu o comando STRCMTCTL. A definição de compromisso contém informações relativas aos recursos que estão sendo alterados sob controle de compromisso dentro desse trabalho. As informações de controle de compromisso na definição de compromisso são mantidas pelo sistema à medida que os recursos de compromisso mudam. A definição de compromisso é encerrada usando o comando End Commitment Control (ENDCMTCTL). Para obter mais informações sobre o controle de confirmação, consulte o tópico Backup e recuperação.

## Definições de compromisso e grupos de ativação

Várias definições de compromisso podem ser iniciadas e usadas por programas em execução em um trabalho. Cada definição de compromisso para um trabalho identifica uma transação separada que tem recursos associados a ela. Esses recursos podem ser comprometidos ou revertidos independentemente de todas as outras definições de compromisso iniciadas para o trabalho.

| **Nota:** Somente programas ILE podem iniciar o controle de compromisso para grupos de ativação que não sejam um | padrão grupo de ativação. Portanto, um trabalho pode usar várias definições de compromisso somente se o trabalho for | executando um ou mais programas ILE.

| Os programas OPM (modelo de programa original) são executados no grupo de ativação padrão de armazenamento de nível único.

| Por padrão, os programas OPM usam a definição de compromisso \*DFTACTRP . Para programas OPM, você | pode usar a definição de compromisso \*JOB especificando CMTSCOPE(\*JOB) no STRCMTCTL

| comando.

| Ao usar o comando Start Commitment Control (STRCMTCTL), você especifica o escopo para um

| definição de compromisso no parâmetro escopo de compromisso (CMTSCOPE). A **margem** para um compromisso | define indica quais programas executados dentro do trabalho usam essa definição de compromisso. O padrão

| escopo para uma definição de compromisso é para o grupo de ativação do programa que emite o STRCMTCTL

| comando. Somente os programas executados dentro desse grupo de ativação usarão essa definição de compromisso, | exceto que os grupos de ativação padrão compartilham uma definição de compromisso. Definições de compromisso que | são definidos para um grupo de ativação são referidos como definições de compromisso no  **nível** do **grupo de ativação**.

| A definição de compromisso iniciada no nível do grupo de ativação para um grupo de ativação padrão é conhecida

| como a definição de compromisso padrão do grupo de ativação (\*DFTACTGRP). Definições de compromisso para muitos | Os níveis de grupo de ativação podem ser iniciados e usados por programas executados em vários grupos de ativação | para um emprego.

Uma definição de compromisso também pode ser definida para o trabalho. Uma definição de compromisso com esse valor de escopo é chamada de  **nível de trabalho** ou definição de compromisso \*JOB . Qualquer programa em execução em um grupo de ativação que não tenha uma definição de compromisso iniciada no nível do grupo de ativação usa a definição de compromisso no nível do trabalho. Isso ocorre se a definição de compromisso no nível do trabalho já tiver sido iniciada por outro programa para o trabalho. Apenas uma única definição de compromisso de nível de trabalho pode ser iniciada para um trabalho.

Para um determinado grupo de ativação, apenas uma única definição de compromisso pode ser usada pelos programas executados nesse grupo de ativação. Os programas executados em um grupo de ativação podem usar a definição de compromisso no nível do trabalho ou no nível do grupo de ativação. No entanto, eles não podem usar as duas definições de compromisso ao mesmo tempo.

Quando um programa executa uma operação de controle de compromisso, o programa não indica diretamente qual definição de compromisso usar para a solicitação. Em vez disso, o sistema determina qual definição de compromisso usar com base em qual grupo de ativação o programa solicitante está sendo executado. Isso é possível porque, a qualquer momento, os programas executados em um grupo de ativação podem usar apenas uma única definição de compromisso.

## Encerrando o controle de compromisso

O controle de compromisso pode ser encerrado para a definição de compromisso no nível do trabalho ou no nível do grupo de ativação usando o comando ENDCMTCTL ( End Commitment Control). O comando ENDCMTCTL indica ao sistema que a definição de compromisso para o grupo de ativação do programa que faz a solicitação deve ser encerrada. O comando ENDCMTCTL encerra uma definição de compromisso para o trabalho. Todas as outras definições de compromisso para o trabalho permanecem inalteradas.

Se a definição de compromisso no nível do grupo de ativação for encerrada, os programas em execução nesse grupo de ativação não poderão mais fazer alterações sob controle de compromisso. Se a definição de compromisso no nível do trabalho for iniciada ou já existir, qualquer nova operação de abertura de arquivo que especifique o controle de compromisso usará a definição de compromisso no nível do trabalho.

Se a definição de compromisso no nível do trabalho for encerrada, qualquer programa em execução no trabalho que estava usando a definição de compromisso no nível do trabalho não poderá mais fazer alterações sob controle de compromisso . Se o controle de compromisso for iniciado novamente com o comando STRCMTCTL, as alterações poderão ser feitas.

Capítulo 11. Escopo de gerenciamento de dados

## Controle de compromisso durante o término do grupo de ativação

Quando as seguintes condições existirem ao mesmo tempo:

v Um grupo de ativação termina v O trabalho não está terminando

o sistema encerra automaticamente uma definição de compromisso em um nível de grupo de ativação. Se ambas as seguintes condições existirem:

v Existem alterações não confirmadas para uma definição de compromisso em um nível de grupo de ativação v O grupo de ativação está terminando normalmente

o sistema executa uma operação de confirmação implícita para a definição de compromisso antes de terminar a definição de compromisso. Caso contrário, se uma das seguintes condições existir:

v O grupo de ativação está terminando de forma anormal v O sistema encontrou erros ao fechar todos os arquivos abertos sob controle de compromisso com escopo para o grupo de ativação

uma operação de reversão implícita é executada para a definição de compromisso no nível do grupo de ativação antes de ser encerrada. Como o grupo de ativação termina de forma anormal, o sistema atualiza o objeto de notificação com a última operação de confirmação bem-sucedida. A confirmação e a reversão são baseadas em alterações pendentes. Se não houver alterações pendentes, não haverá reversão, mas o objeto de notificação ainda será atualizado. Se o grupo de ativação terminar anormalmente com alterações pendentes, o sistema implicitamente reverterá as alterações. Se o grupo de ativação terminar normalmente com alterações pendentes, o sistema confirmará implicitamente as alterações.

Uma operação de confirmação implícita ou uma operação de reversão nunca é executada durante o processamento final do grupo de ativação para as definições de compromisso \*JOB ou \*DFTATCTRP . Isso ocorre porque as definições de compromisso \*JOB e \*DFTACTGRP nunca são encerradas devido ao término de um grupo de ativação. Em vez disso, essas definições de compromisso são explicitamente encerradas por um comando ENDCMTCTL ou terminadas pelo sistema quando o trabalho termina.

O sistema fecha automaticamente todos os arquivos com escopo para o grupo de ativação quando o grupo de ativação termina. Isso inclui todos os arquivos de banco de dados com escopo para o grupo de ativação aberto sob controle de compromisso . A operação de fechamento para qualquer arquivo desse tipo ocorre antes de qualquer operação de confirmação implícita executada para a definição de compromisso no nível do grupo de ativação. Portanto, todos os registros que residem em um buffer de E/S são primeiro forçados ao banco de dados antes que qualquer operação de confirmação implícita seja executada.

Como parte da operação implícita de confirmação ou reversão , o sistema chama o programa de confirmação e saída de reversão da API para cada recurso de confirmação de API. Cada recurso de compromisso de API deve ser associado à definição de compromisso no nível do grupo de ativação. Depois que o programa de saída de confirmação e reversão da API é chamado, o sistema remove automaticamente o recurso de confirmação da API.

| O objeto de notificação será atualizado se as seguintes condições existirem:

v Uma operação de reversão implícita é executada para uma definição de compromisso que está sendo encerrada porque um grupo de ativação está sendo encerrado v Um objeto de notificação é definido para o compromisso definição

# Capítulo 12. Interfaces de programação de aplicativos vinculáveis ILE

As interfaces de programação de aplicativos vinculáveis ao ILE (APIs vinculáveis) são uma parte importante do ILE. Em alguns casos, eles fornecem uma função adicional além da fornecida por uma linguagem específica de alto nível. Por exemplo, nem todos os HLLs oferecem meios intrínsecos para manipular o armazenamento dinâmico. Nesses casos, você pode complementar uma função HLL usando APIs vinculáveis específicas . Se o seu HLL fornecer a mesma função que uma API vinculável específica , use a específica do HLL.

As APIs vinculáveis são independentes de HLL. Isso pode ser útil para aplicativos de linguagem mista. Por exemplo, se você usar apenas APIs vinculáveis de gerenciamento de condições com um aplicativo de linguagem mista, você terá semântica uniforme de manipulação de condição para esse aplicativo. Isso torna o gerenciamento de condições mais consistente do que ao usar vários manipuladores de condição específicos de HLL.

As APIs vinculáveis fornecem uma ampla gama de funções, incluindo:

Grupo de ativação e gerenciamento de fluxo de controle

Gerenciamento de condições

Manipulação de data e hora

Gerenciamento dinâmico de tela

Funções matemáticas

Tratamento de mensagens

Gerenciamento de chamadas de programa ou procedimento e acesso ao descritor operacional

Depurador de origem

Gerenciamento de armazenamento

Para obter informações de referência sobre as APIs vinculáveis do ILE, consulte o tópico API na categoria Programação do Centro de Informações do IBM i.

# APIs vinculáveis ILE disponíveis

A maioria das APIs vinculáveis está disponível para qualquer HLL compatível com ILE. O ILE fornece as seguintes APIs vinculáveis:

#### Grupo de ativação e fluxo de controle de APIs vinculáveis extremidade anormal (CEE4ABN)

Encontrar um limite de controle (CEE4FCB)

Registrar o Procedimento de Saída do Grupo de Ativação (CEE4RAGE)

Registrar o Procedimento de Saída do Usuário de Entrada de Pilha de Chamadas (CEERTX) Sinalizar a Condição de Terminação-Iminente (CEETREC)

Cancelar o registro do procedimento de saída do usuário de término de pilha de chamadas (CEEUTX)

#### APIs vinculáveis de gerenciamento de condições

Construir um Token de Condição (CEENCOD)

Decompor um Token de Condição (CEEDCOD)

Manipular uma condição (CEE4HC)

Mover o cursor de retomada para um ponto de retorno (CEEMRCR)

Registrar um manipulador de condição escrito pelo usuário (CEEHDLR)

Recuperar a versão do ILE e o ID da plataforma (CEEGPID)

Retornar o número de invocação relativo (CEE4RIN)

Sinalizar uma Condição (CEESGL)

Cancelar o registro de um manipulador de condição de usuário (CEEHDLU)

#### APIs vinculáveis de data e hora

Calcular o dia da semana a partir da data de Lilian (CEEDYWK)

Converter data para o formato Lilian (CEEDAYS)

© Direitos autorais IBM Corp. 1997, 2010

Converter inteiros em segundos (CEEISEC)

Converter data Lilian para formato de caractere (CEEDATE)

Converter segundos em carimbo de data/hora de caractere (CEEDATM)

Converter segundos em inteiros (CEESECI)

Converter carimbo de data/hora em número de segundos (CEESECS)

Obter Hora Média Atual de Greenwich (CEEGMT)

Obter Hora Local Atual (CEELOCT)

Obter deslocamento do tempo universal coordenado para a hora local (CEEUTCO)

Obtenha o Tempo Universal Coordenado (CEEUTC)

Século de Consulta (CEEQCEN)

Retornar cadeias de caracteres de data e hora padrão para país ou região (CEEFMDT)

Retornar Cadeia de Dados Padrão para País ou Região (CEEFMDA)

Retornar Cadeia de Caracteres de Tempo Padrão para País ou Região (CEEFMTM) Definido Século (CEESCEN)

#### APIs vinculáveis de matemática

O x no nome de cada API vinculável à matemática refere-se a um dos seguintes tipos de dados:

**I**  inteiro binário de 32 bits

**Número de** ponto flutuante único de 32 bits S

**Número de**  ponto flutuante duplo de 64 bits D

**Número** único de complexo flutuante de 32 bits T (as partes reais e imaginárias têm 32 bits)

**E**  número duplo de complexo flutuante de 64 bits (as partes reais e imaginárias têm 64 bits de comprimento)

Função absoluta (CEESxABS)

Arccosseno (CEESxACS)

Arcseno (CEESxASN)

Arctangente (CEESxATN)

Arctangente2 (CEESxAT2)

Conjugado de Complexo (CEESxCJG)

Coseno (CEESxCOS)

Cotangente (CEESxCTN)

Função de erro e seu complemento (CEESxERx)

Base exponencial e (CEESxEXP)

Exponenciação (CEESxXPx)

Fatorial (CEE4SIFAC)

Divisão Complexa Flutuante (CEESxDVD)

Função Gama de Multiplicação de Complexo Flutuante (CEESxMLT) (CEESxGMA)

Arctangente hiperbólica (CEESxATH)

Cosino hiperbólico (CEESxCSH) Seno hiperbólico (CEESxSNH)

Tangente hiperbólica (CEESxTNH)

Parte imaginária do complexo (CEESxIMG)

Função Log Gamma (CEESxLGM)

Logaritmo Base 10 (CEESxLG1)

Logaritmo Base 2 (CEESxLG2)

Logaritmo Base e (CEESxLOG)

Aritmética modular (CEESxMOD)

Inteiro mais próximo (CEESxNIN)

Número inteiro mais próximo (CEESxNWN)

Diferença positiva (CEESxDIM)

Seno (CEESxSIN)

Raiz quadrada (CEESxSQT)

Tangente (CEESxTAN)

Transferência de Sinal (CEESxSGN)

Truncamento (CEESxINT)

API vinculável matemática adicional:

Geração Básica de Números Aleatórios (CEERAN0)

#### APIs vinculáveis de manipulação de mensagens

Despachar uma Mensagem (CEEMOUT)

Obter uma Mensagem (CEEMGET)

Obter, formatar e despachar uma mensagem (CEEMSG)

#### APIs vinculáveis de chamada de programa ou procedimento

Obter informações de cadeia de caracteres (CEEGSI)

Recuperar informações do descritor operacional (CEEDOD)

Teste para argumento omitido (CEETSTA)

#### APIs vinculáveis do depurador de origem

Permitir que um programa emita instruções de depuração (QteSubmitDebugCommand)

Habilitar uma sessão para usar o depurador de origem (QteStartSourceDebug)

Mapear posições de uma visualização para outra (QteMapViewPosition)

Registrar uma exibição de um módulo (QteRegisterDebugView) Remover uma exibição de um módulo (QteRemoveDebugView)

Recuperar os atributos da sessão de depuração de origem (QteRetrieveDebugAttribute)

Recuperar a lista de módulos e exibições de um programa (QteRetrieveModuleViews)

Recuperar a posição em que o programa parou (QteRetrieveStoppedPosition)

Recuperar texto de origem do modo de exibição especificado (QteRetrieveViewText)

Definir os atributos da sessão de depuração de origem (QteSetDebugAttribute)

Tirar um trabalho do modo de depuração (QteEndSourceDebug)

#### APIs vinculáveis de gerenciamento de armazenamento

Criar pilha (CEECRHP)

Definir Heap Allocation Strategy (CEE4DAS)

Descartar Pilha (CEEDSHP)

Armazenamento Gratuito (CEEFRST)

Obter armazenamento de pilha (CEEGTST)

Mark Heap (CEEMKHP)

Realocar Armazenamento (CEECZST)

Pilha de liberação (CEERLHP)

# APIs vinculáveis do Gerenciador de Tela Dinâmica

As APIs vinculáveis do gerenciador dinâmico de tela (DSM) são um conjunto de interfaces de E/S de tela que fornecem uma maneira dinâmica de criar e gerenciar telas de exibição para as linguagens de alto nível do ILE.

As APIs do DSM se enquadram nos seguintes grupos funcionais:

#### v Serviços de baixo nível

As APIs de serviços de baixo nível fornecem uma interface direta para os comandos de fluxo de dados 5250. As APIs são usadas para consultar e manipular o estado da tela de exibição; para criar, consultar e manipular buffers de entrada e comando que interagem com a tela de exibição; e para definir campos e gravar dados na tela de exibição. v **Serviços do Windows**

As APIs de serviços de janela são usadas para criar, excluir, mover e redimensionar janelas; e para gerenciar várias janelas simultaneamente durante uma sessão.

#### v Serviços de sessão

As APIs de serviços de sessão fornecem uma interface de paginação geral que pode ser usada para criar, consultar e manipular sessões; e para executar operações de entrada e saída para sessões.

Capítulo 12. Interfaces de programação de aplicativos vinculáveis ILE

Para obter informações sobre as APIs vinculáveis do DSM , consulte a coleção de tópicos da API na categoria Programação do Centro de Informações do IBM i.

# Capítulo 13. Técnicas avançadas de otimização

Este capítulo descreve as seguintes técnicas que você pode usar para otimizar seus programas ILE e programas de serviço:

v "Adaptive Code Generation" na página 143 v "Advanced Argument Optimization" na página 136 v "Interprocedural analysis (IPA)" na página 130 v "Licensed Internal Code Options" na página 138 v "Program Profiling "

# Criação de perfil do programa

A criação de perfil de programa é uma técnica avançada de otimização para reordenar procedimentos, ou codificar dentro de procedimentos, em programas ILE e programas de serviço com base em dados estatísticos coletados durante a execução do programa. Essa reordenação pode melhorar a utilização do cache de instruções e reduzir a paginação exigida pelo programa, melhorando assim o desempenho. O comportamento semântico do programa não é afetado pela criação de perfil do programa.

A melhoria de desempenho realizada pela criação de perfil do programa depende do tipo de aplicativo. De um modo geral, você pode esperar mais melhorias de programas que passam a maior parte do tempo no próprio código do aplicativo, em vez de gastar tempo no tempo de execução ou fazendo processamento de entrada/saída. O desempenho do código do programa produzido ao aplicar dados de perfil depende do tradutor otimizador identificar corretamente as partes mais importantes do programa durante o uso típico. Portanto, é importante coletar dados de perfil ao executar as tarefas que serão executadas pelos usuários finais e usar dados de entrada semelhantes aos esperados em o ambiente no qual o programa será executado.

A criação de perfil de programa está disponível somente para programas ILE e programas de serviço que atendam às seguintes condições:

v Os programas foram criados especificamente para V4R2M0 ou versões posteriores.

v Se os programas foram criados para uma versão anterior à V5R2M0, a versão de destino do programa deve ser a mesma que a versão atual do sistema.

v Os programas foram compilados usando um nível de otimização de \*FULL (30) ou superior. Em sistemas V5R2M0 e posteriores, módulos acoplados com nível inferior a 30 de otimização são permitidos, mas não participam da criação de perfil do aplicativo.

Nota: Devido aos requisitos de otimização, você deve depurar totalmente seus programas antes de usar a criação de perfil do programa.

## Tipos de criação de perfil

Você pode criar o perfil de seus programas das duas maneiras a seguir:

v Ordem de bloqueio

v Ordem processual e ordem de bloqueio

A criação de perfil de ordem de bloco registra o número de vezes que cada lado de uma ramificação condicional é usado. Quando os dados de criação de perfil de ordem de bloco são aplicados a um programa, uma variedade de otimizações baseadas em perfil são executadas dentro dos procedimentos pelo tradutor de otimização. Para uma dessas otimizações, o código é ordenado para que os caminhos de código executados com mais frequência em um procedimento sejam contíguos dentro do objeto de programa. Essa reordenação melhora o desempenho utilizando componentes do processador, como o cache de instruções e a unidade de pré-busca de instruções de forma mais eficaz.

© Direitos autorais IBM Corp. 1997, 2010

A criação de perfil de ordem de procedimento registra o número de vezes que cada procedimento chama outro procedimento dentro do programa. Os procedimentos dentro do programa são reordenados para que os procedimentos mais frequentemente chamados sejam empacotados juntos. Essa reordenação melhora o desempenho, reduzindo a paginação de memória.

Mesmo que você possa optar por aplicar apenas o perfil de ordem de bloco ao seu programa, é recomendável que você aplique os dois tipos para obter os maiores ganhos de desempenho.

## Como criar o perfil de um programa

A criação de perfil de um programa é um processo de cinco etapas:

1. Habilite o programa para coletar dados de criação de perfil.
2. Inicie a coleção de criação de perfil de programa no sistema com o comando Iniciar Criação de Perfil de Programa (STRPGMPRF).
3. Colete dados de criação de perfil executando o programa por meio de seus caminhos de código de alto uso. Como a criação de perfil do programa usa dados estatísticos coletados durante a execução do programa para executar essas otimizações, é fundamental que esses dados sejam coletados em usos típicos do seu aplicação.
4. Encerre a coleção de criação de perfil de programa no sistema com o comando End Program Profiling (ENDPGMPRF).
5. Aplique os dados de criação de perfil coletados ao programa solicitando que o código seja reordenado para um desempenho ideal com base nos dados de criação de perfil coletados.

### Habilitando o programa para coletar dados de criação de perfil

Um programa é habilitado para coletar dados de criação de perfil se pelo menos um dos módulos vinculados ao programa estiver habilitado para coletar dados de criação de perfil. Habilitar um programa para coletar dados de criação de perfil pode ser feito alterando um ou mais objetos \*MODULE para coletar dados de criação de perfil e , em seguida, criando ou atualizando o programa com esses módulos, ou alterando o programa depois que ele é criado para coletar dados de criação de perfil. Ambas as técnicas resultam em um programa com módulos acoplados habilitados para coletar dados de criação de perfil.

Dependendo da linguagem ILE que você está usando, pode haver uma opção no comando do compilador para criar o módulo como habilitado para coletar dados de criação de perfil. O comando change module (CHGMOD) também pode ser usado especificando \*COL no parâmetro de dados de criação de perfil (PRFDTA) para alterar qualquer módulo ILE para coletar dados de criação de perfil, desde que o ILE linguagem suporta um nível de otimização de pelo menos \*FULL (30).

Para permitir que um programa colete dados de criação de perfil após a criação por meio dos comandos Programa de Alteração (CHGPGM) ou Programa de Serviço de Alteração (CHGSRVPGM), faça o seguinte para um programa observável :

v Especifique \*COL no parâmetro de dados de criação de perfil (PRFDTA). Essa especificação afeta todos os módulos vinculados no programa que:

* Foram criados para uma versão do V4R2M0 ou posterior. Se você estiver usando um sistema anterior à V5R2M0, o programa deve estar em um sistema no mesmo nível de versão para o qual o programa foi criado para habilitar a criação de perfil coleta de dados. A mesma restrição se aplica aos módulos acoplados.
* Ter um nível de otimização de 30 ou superior. Na V5R2M0 e versões posteriores, quaisquer módulos acoplados com menos de nível de otimização 30 serão permitidos, mas não participarão da criação de perfil do aplicativo.

**Nota:** Um programa habilitado para coletar dados de criação de perfil de aplicativo em um sistema com liberação anterior ao V5R2M0 pode ter esses dados aplicados em um sistema V5R2M0 ou posterior, mas os resultados podem não ser ideal. Se você pretende aplicar dados de criação de perfil ou usar o programa resultante em um sistema V5R2M0 ou posterior, você deve habilitar ou reativar a coleta de dados de criação de perfil para o programa em um sistema V5R2M0 ou posterior.

Habilitar um módulo ou programa para coletar dados de criação de perfil requer que o objeto seja recriado. Portanto, o tempo necessário para permitir que um módulo ou programa colete dados de criação de perfil é comparável ao tempo necessário para forçar a recriação do objeto (parâmetro FRCCRT). Além disso, o tamanho do objeto será maior devido às instruções extras da máquina geradas pelo tradutor de otimização.

Depois de habilitar um programa ou módulo para coletar dados de criação de perfil, a observabilidade dos dados de criação não poderá ser removida até que ocorra uma das seguintes situações: v Os dados de criação de perfil coletados sejam aplicados a o programa. v O programa ou módulo é alterado para que ele não possa coletar dados de criação de perfil.

Usar o DSPMOD (Display Module), o Display Program ( DSPPGM) ou o Display Service Program

(DSPSRVPGM) , especificando DETAIL(\*BASIC), para determinar se um módulo ou programa está habilitado para coletar dados de criação de perfil. Para programas ou programas de serviço , use a opção 5 (descrição de exibição) do DETAIL(\*MODULE) para determinar qual dos módulos acoplados está habilitado para coletar dados de criação de perfil. Consulte o tópico "Como saber se um programa ou módulo está com perfil ou habilitado para coleção" na página 130 para obter mais detalhes.

**Nota:** Se um programa já tiver dados de criação de perfil coletados (os dados estatísticos coletados enquanto o programa está em execução), esses dados serão limpos quando um programa for reativado para coleta dados de criação de perfil. Consulte "Gerenciando programas habilitados para coletar dados de criação de perfil" na página 128 para obter detalhes.

### Coletar dados de criação de perfil

A criação de perfil do programa deve ser iniciada na máquina em que um programa habilitado para coletar dados de criação de perfil deve ser executado para que esse programa atualize as contagens de dados de criação de perfil . Isso permite que aplicativos grandes e de longa duração sejam iniciados e tenham acesso a um estado estável antes de coletar dados de criação de perfil. Isso lhe dá controle sobre quando a coleta de dados ocorre.

Use o comando Iniciar Criação de Perfil de Programa (STRPGMPRF) para iniciar a criação de perfil de programa em uma máquina. Para encerrar a criação de perfil de programa em uma máquina, use o comando End Program Profiling (ENDPGMPRF). Ambos os comandos são enviados com a autoridade pública de \*EXCLUDE. A criação de perfil do programa é encerrada implicitamente quando uma máquina é IPLed.

Depois que a criação de perfil do programa for iniciada, qualquer programa ou programa de serviço executado que também esteja habilitado para coletar dados de criação de perfil atualizará suas contagens de dados de criação de perfil. Isso acontecerá independentemente de o programa ter sido ativado ou não antes da emissão do comando STRPGMPRF.

Se o programa no qual você está coletando dados de criação de perfil puder ser chamado por vários trabalhos na máquina, as contagens de dados de criação de perfil serão atualizadas por todos esses trabalhos. Se isso não for desejável, uma cópia duplicada do programa deve ser feita em uma biblioteca separada e essa cópia deve ser usada em vez disso.

**Anotações:**

1. Quando a criação de perfil do programa é iniciada em uma máquina, as contagens de dados de criação de perfil são incrementadas enquanto um programa habilitado para coletar dados de criação de perfil está em execução. Portanto, é possível que contagens de dados de criação de perfil "obsoletas" estejam sendo adicionadas se esse programa tiver sido executado anteriormente sem limpar essas contagens posteriormente. Você pode forçar as contagens de dados de criação de perfil a serem limpas de várias maneiras. Consulte "Gerenciando programas habilitados para coletar dados de criação de perfil" na página 128 para obter detalhes.
2. As contagens de dados de criação de perfil não são gravadas no DASD cada vez que são incrementadas porque isso causaria uma degradação muito grande no tempo de execução do programa. As contagens de dados de criação de perfil só são gravadas no DASD quando o programa é naturalmente paginado. Para garantir que as contagens de dados de criação de perfil sejam gravadas no DASD, use o comando Limpar Pool (CLRPOOL) para limpar o pool de armazenamento no qual o programa está sendo executado.

### Aplicando os dados de criação de perfil coletados

A aplicação dos dados de criação de perfil coletados faz o seguinte:

1. Instrui a máquina a usar os dados de criação de perfil coletados para reordenar procedimentos (dados de criação de perfil de ordem de procedimento ) no programa para obter o desempenho ideal.
2. Instrui a máquina a usar os dados de criação de perfil coletados (dados básicos de criação de perfil de bloco) para reordenar o código dentro dos procedimentos no programa para obter o desempenho ideal.
3. Remove as instruções da máquina do programa que foram adicionadas anteriormente quando o programa foi habilitado para coletar dados de criação de perfil. O programa não pode mais coletar dados de perfil.
4. Armazena os dados de criação de perfil coletados no programa como dados observáveis: v \*BLKORD (observabilidade básica de perfil de bloco) v \*PRCORD ( observabilidade de perfil de ordem de procedimento)

Uma vez que os dados coletados tenham sido aplicados ao programa, eles não poderão ser aplicados novamente. Para aplicar dados de criação de perfil novamente, você deve seguir as etapas descritas em "Como criar o perfil de um programa" na página 126. **Todos os** dados de criação de perfil  **aplicados anteriormente** **são descartados quando um programa é habilitado para coletar dados de criação de perfil.**

Se você quiser aplicar os dados que você já coletou novamente, convém fazer uma cópia do programa antes de aplicar dados de criação de perfil. Isso pode ser desejável se você estiver experimentando os benefícios derivados de cada tipo de criação de perfil (ordem de bloco ou bloco e procedimento ordenado).

Para aplicar dados de criação de perfil, use o comando Programa de Alteração (CHGPGM) ou Programa de Serviço de Alteração (CHGSRVPGM ). Para o parâmetro de dados de criação de perfil (PRFDTA), especifique:

v Dados de criação de perfil de ordem de bloco (\*APYBLKORD)

v Dados de criação de perfil de ordem de bloco e procedimento (\*APYALL) ou (\*APYPRCORD) A IBM recomenda o uso de \*APYALL.

A aplicação de dados de criação de perfil ao programa cria e salva duas formas adicionais de observabilidade com o programa. Você pode remover essas observabilidades adicionais usando os comandos Programa de Alteração (CHGPGM) e Programa de Serviço de Alteração (CHGSRVPGM).

v \*A observabilidade do BLKORD é implicitamente adicionada quando os dados de criação de perfil de ordem de bloco são aplicados ao programa. Isso permite que a máquina preserve os dados de criação de perfil de ordem de bloco aplicados para o programa nos casos em que o programa é recriado.

v A aplicação de dados de criação de perfil de ordem de procedimento ao programa adiciona implicitamente \*PRCORD e \*BLKORD observabilidade. Isso permite que a máquina preserve os dados de criação de perfil de ordem de procedimento aplicados para o programa nos casos em que o programa é recriado ou atualizado.

Por exemplo, você aplica dados de criação de perfil de ordem de bloco ao seu programa e, posteriormente, remove a observabilidade \*BLKORD . O programa ainda é de ordem de bloco perfilado. No entanto, qualquer alteração que faça com que seu programa seja recriado também fará com que ele não seja mais bloqueado com perfil de ordem.

## Gerenciando programas habilitados para coletar dados de criação de perfil

Alterar um programa habilitado para coletar dados de criação de perfil usando os comandos Programa de Alteração (CHGPGM) ou Programa de Serviço de Alteração (CHGSRVPGM ) fará com que implicitamente as contagens de dados de criação de perfil ser zerado se a alteração exigir que o programa seja recriado. Por exemplo, se você alterar um programa habilitado para coletar dados de criação de perfil do nível de otimização \*FULL para o nível de otimização 40, todos os dados de criação de perfil coletados serão ser implicitamente esclarecido. Isso também é verdadeiro se um programa habilitado para coletar dados de criação de perfil for restaurado e FRCOBJCVN(\*SIM \*ALL) for especificado no comando Restore Object (RSTOBJ).

Da mesma forma , a atualização de um programa habilitado para coletar dados de criação de perfil usando os comandos Programa de Atualização (UPDPGM) ou Programa de Serviço de Atualização (UPDSRVPGM) causará implicitamente dados de criação de perfil contagens a serem limpas se o programa resultante ainda estiver habilitado para coletar dados de criação de perfil. Por exemplo, o programa P1 contém os módulos M1 e M2. O módulo M1 vinculado em P1 está habilitado para coletar dados de criação de perfil, mas o módulo M2 não. Contanto que um dos módulos esteja habilitado, a atualização do programa P1 com o módulo M1 ou M2 resultará em um programa que ainda está habilitado para coletar dados de criação de perfil. Todas as contagens de dados de criação de perfil serão limpas. No entanto, se o módulo M1 for alterado para não mais ser habilitado para coletar dados de criação de perfil especificando \*NOCOL no parâmetro de dados de criação de perfil (PRFDTA) do Comando Alterar módulo (CHGMOD), atualizar o programa P1 com M1 fará com que o programa P1 não seja mais habilitado para coletar dados de criação de perfil.

Você pode limpar explicitamente as contagens de criação de perfil do programa especificando a opção \*CLR na criação de perfil

parâmetro data (PRFDTA) do Programa de Alteração (CHGPGM) ou do Programa de Serviço de Alteração

(CHGSRVPGM) comandos. Observe que o programa não deve ser ativado para usar a opção \*CLR.

Se você não quiser mais que o programa colete dados de criação de perfil, você pode executar uma das seguintes ações:

v Especifique \*NOCOL no parâmetro de dados de criação de perfil (PRFDTA) do Programa de Alterações (CHGPGM)

comando.

v Especifique \*NOCOL no parâmetro de dados de criação de perfil (PRFDTA) do comando Change Service Program (CHGSRVPGM).

Qualquer ação altera o programa de volta ao estado antes de coletar dados de criação de perfil. Você também pode alterar o valor PRFDTA dos módulos para \*NOCOL com o comando CHGMOD ou recompilando os módulos e revinculando os módulos no programa.

## Gerenciando programas com dados de criação de perfil aplicados a eles

Se um programa que tem dados de criação de perfil aplicados for alterado usando os comandos Programa de Alteração (CHGPGM) ou Programa de Serviço de Alteração (CHGSRVPGM ), você perderá os dados de criação de perfil aplicados se ambas as condições são verdadeiras: v A alteração requer que o programa seja recriado.

**Nota:** O nível de otimização de um programa que tem dados de criação de perfil aplicados não pode ser alterado para ser inferior ao nível de otimização 30. Isso ocorre porque os dados de criação de perfil dependem do nível de otimização. v A observabilidade de criação de perfil necessária foi removida.

Além disso, todos os dados de criação de perfil aplicados serão perdidos se a solicitação de alteração for para permitir que o programa colete dados de criação de perfil , independentemente de a observabilidade de criação de perfil ter sido removida ou não. Essa solicitação resultará em um programa habilitado para coletar dados de criação de perfil.

Aqui estão alguns exemplos:

v O Programa A tem dados de criação de perfil de ordem de procedimento e ordem de bloco aplicados. \*A observabilidade do BLKORD tem

foi removido do programa, mas \*PRCORD observabilidade não. Um comando CHGPGM é executado para alterar o atributo de coleta de desempenho do programa A, que também requer que o programa seja recriado. Essa solicitação de alteração fará com que o programa A não seja mais perfilado como ordem de bloqueio. No entanto, os dados de criação de perfil de ordem de procedimento ainda serão aplicados.

v O Programa A tem dados de criação de perfil de ordem de procedimento e ordem de bloco aplicados. \*BLKORD e \*PRCORD

a observabilidade foi removida do programa. Um comando CHGPGM é executado para alterar o atributo de perfil de usuário do programa A, que também requer que o programa seja recriado. Essa solicitação de alteração fará com que o programa A não seja mais a ordem de bloqueio ou a ordem de procedimento perfilada. O Programa A voltará ao estado anterior à aplicação dos dados de criação de perfil.

v O Programa A tem dados de criação de perfil de ordem de bloco aplicados. \*A observabilidade BLKORD foi removida do programa. Um comando CHGPGM é executado para alterar o texto do programa, que **não** requer que o programa seja recriado. Após essa alteração, o programa A ainda será perfilado como ordem de bloqueio.

v O Programa A tem dados de criação de perfil de ordem de procedimento e ordem de bloco aplicados. Isso não remove a observabilidade \*PRCORD e \*BLKORD do programa. Execute um comando CHGPGM para permitir que o programa colete dados de criação de perfil (isso recria o programa). Isso faz com que o programa A não seja mais a ordem de bloqueio ou a ordem de procedimento perfilada. Isso deixa o programa em um estado como se os dados de criação de perfil nunca tivessem sido aplicados. Isso permite que o programa colete dados de criação de perfil com todas as contagens de dados de criação de perfil limpas.

Um programa que teve dados de criação de perfil aplicados a ele (\*APYALL, \*APYBLKORD ou \*APYPRCORD) não pode ser imediatamente alterado para um programa sem perfil especificando PRFDTA(\*NOCOL) no CHGPGM ou CHGSRVPGM comandos. Este recurso destina-se a ser uma rede de segurança para evitar a perda acidental de dados de criação de perfil. Se isso é realmente o que se pretende, o programa deve primeiro ser alterado para PRFDTA(\*COL), removendo efetivamente os dados de criação de perfil existentes e, em seguida, alterado para PRFDTA(\*NOCOL).

## Como saber se um programa ou módulo está com perfil ou habilitado para coleta

Use os comandos Programa de exibição (DSPPGM) ou Programa de serviço de exibição (DSPSRVPGM), especificando DETAIL(\*BASIC), para determinar o atributo de dados de criação de perfil de programa de um programa. O valor de "Dados de criação de perfil" será um dos seguintes valores: v \*NOCOL - O programa não está habilitado para coletar dados de criação de perfil.

v \*COL - Um ou mais módulos do programa estão habilitados para coletar dados de criação de perfil. Esse valor não indica se os dados de criação de perfil foram realmente coletados.

v \*APYALL - Os dados de criação de perfil de ordem de bloco e ordem de procedimento são aplicados a este programa. A coleta de dados de criação de perfil não está mais habilitada.

v \*APYBLKORD - Os dados de criação de perfil de ordem de bloco são aplicados aos procedimentos de um ou mais módulos acoplados neste programa. Isso se aplica somente aos módulos acoplados que foram habilitados anteriormente para coletar dados de criação de perfil. A coleta de dados de criação de perfil não está mais habilitada.

v \*APYPRCORD- Os dados de criação de perfil do programa de ordem de procedimento são aplicados a este programa. A coleta de dados de criação de perfil não está mais habilitada.

Para que apenas a criação de perfil de ordem de procedimento seja aplicada a um programa:

v Primeiro perfilá-lo especificando \*APYALL ou \*APYPRCORD (que é o mesmo que \*APYALL). v Em seguida, remova a observabilidade \*BLKORD e recrie o programa.

Para exibir o atributo de dados de criação de perfil de programa de um módulo vinculado dentro do programa, use DSPPGM ou DSPSRVPGM DETAIL(\*MODULE). Especifique a opção 5 nos módulos vinculados ao programa para ver o valor desse parâmetro no nível do módulo. O valor de "Dados de criação de perfil " será um dos seguintes valores: v \*NOCOL - Este módulo acoplado não está habilitado para coletar dados de criação de perfil.

v \*COL - Este módulo acoplado está habilitado para coletar dados de criação de perfil. Esse valor não indica se os dados de criação de perfil foram realmente coletados.

v \*APYBLKORD - Os dados de criação de perfil de ordem de bloco são aplicados a um ou mais procedimentos deste módulo vinculado. A coleta de dados de criação de perfil não está mais habilitada.

Além disso, DETAIL(\*MODULE) exibe os campos a seguir para fornecer uma indicação do número de procedimentos afetados pelo atributo de dados de criação de perfil do programa.

v Número de procedimentos - Número total de procedimentos no módulo.

v Número de procedimentos bloco reordenado - O número de procedimentos neste módulo que são blocos básicos reordenados.

v Número de procedimentos ordem de bloco medidos - Número de procedimentos neste módulo vinculado que tinham dados de criação de perfil de ordem de bloco coletados quando os dados de criação de perfil de ordem de bloco foram aplicados. Quando o benchmark foi executado, pode ser o caso de não ter sido coletado nenhum dado para um procedimento específico porque o procedimento não foi executado no benchmark. Assim, essa contagem reflete o número real de procedimentos que foram executados com o benchmark.

Use o comando DSPMOD para determinar o atributo de criação de perfil de um módulo. O valor de "Dados de criação de perfil" será um dos seguintes. Ele nunca mostrará \*APYBLKORD porque os dados básicos do bloco podem ser aplicados apenas a módulos vinculados a um programa, nunca a módulos autônomos. v \*NOCOL - o módulo não está habilitado para coletar dados de perfil. v \*COL - módulo está habilitado para coletar dados de perfil.

# Análise interprocessual (IPA)

Este tópico fornece uma visão geral do processamento de IPA ( Análise Interprocessual) que está disponível por meio da opção IPA nos comandos CRTPGM e CRTSRVPGM.

Em tempo de compilação, o tradutor otimizador realiza análises intraprocessuais e interprocessuais. A análise intraprocessual é um mecanismo para executar a otimização para cada função dentro de uma unidade de compilação, usando apenas as informações disponíveis para essa função e compilação. unidade. A análise interprocessual é um mecanismo para executar a otimização através dos limites da função. O tradutor otimizador realiza análise interprocessual, mas apenas dentro de uma unidade de compilação. A análise interprocessual executada pela opção do compilador IPA melhora o limitado

análise interprocessual descrita acima. Quando você executa a análise interprocedural por meio da opção IPA, o IPA executa otimizações em todo o programa. Ele também executa otimizações não disponíveis em tempo de compilação com o tradutor de otimização. O tradutor otimizador ou a opção IPA executa os seguintes tipos de otimizações:

v *Inlining em todas as unidades de compilação.*  O inlining substitui certas chamadas de função pelo código real da função. O inlining não apenas elimina a sobrecarga da chamada, mas também expõe toda a função ao chamador e, portanto, permite que o compilador otimize melhor seu código. .

v  *Particionamento de programas.*  O particionamento de programas melhora o desempenho reordenando funções para explorar a localidade de referência. O particionamento coloca funções que chamam umas às outras com frequência em maior proximidade na memória. Para obter mais informações sobre particionamento de programas, consulte "Partições criadas pelo IPA" na página 135.

v *Coalescência de variáveis* globais*.*  O compilador coloca variáveis globais em uma ou mais estruturas e acessa as variáveis calculando os deslocamentos desde o início das estruturas. Isso reduz o custo do acesso variável e explora a localidade dos dados.

v  *Endireitamento de código.*  O endireitamento de código simplifica o fluxo do seu programa.

v *Eliminação de código inacessível .*  A eliminação de código inacessível remove o código inacessível dentro de uma função.

v *Chamada*  de  *poda gráfica de funções* *inacessíveis.*  A remoção do gráfico de chamadas de funções inacessíveis remove o código que está 100% embutido ou nunca mencionado.

v Propagação  *constante* *intraprocessual* e  *propagação de conjuntos.*  O IPA propaga constantes de ponto flutuante e inteiro para seus usos e calcula expressões constantes em tempo de compilação. Além disso, os usos variáveis que são conhecidos por serem uma das várias constantes podem resultar na dobra de condicionais e interruptores.

v *Análise* de  *alias de ponteiro intraprocessual.*  O IPA rastreia as definições de ponteiro para seus usos, resultando em informações mais refinadas sobre locais de memória que uma referência de ponteiro pode usar ou definir. Isso permite que outras partes do compilador otimizem melhor o código em torno de tais desreferências. O IPA rastreia dados e definições de ponteiro de função. Quando um ponteiro só pode se referir a um único local ou função de memória, o IPA o regrava para ser uma referência explícita ao local ou função da memória.

v *Propagação de cópia intraprocessual.*  O IPA propaga expressões e define algumas variáveis para os usos da variável. Isso cria oportunidades adicionais para o dobramento de expressões constantes. Ele também elimina cópias variáveis redundantes.

v *Código intraprocessual inacessível e eliminação de armazenamento.*  O IPA remove definições de variáveis que ele não pode alcançar, juntamente com a computação que alimenta a definição.

v *Conversão* de argumentos de *referência (endereço) em* *argumentos de valor.*  O IPA converte argumentos de referência (endereço) em argumentos de valor quando o parâmetro formal não é escrito no procedimento chamado.

v *Conversão de*  variáveis *estáticas em variáveis automáticas (pilha).*  O IPA converte variáveis estáticas em variáveis automáticas (pilha) quando seu uso é limitado a uma única chamada de procedimento.

O tempo de execução para o código que é otimizado usando IPA é normalmente mais rápido do que para o código otimizado somente em tempo de compilação. Nem todos os aplicativos são adequados para otimização de IPA, no entanto, e os ganhos de desempenho que são obtidos com o uso do IPA variam . Para determinadas aplicações, o desempenho da aplicação pode não melhorar ao utilizar a análise interprocessual. Na verdade, em alguns casos raros, o desempenho do aplicativo pode realmente degradar quando você usa a análise interprocessual. Se isso ocorrer, sugerimos que você não use a análise interprocessual. A melhoria de desempenho realizada pela análise interprocessual depende do tipo de aplicação. Os aplicativos que provavelmente mostrarão ganhos de desempenho são aqueles que têm as seguintes características: v Contêm um grande número de funções v Contêm um grande número de unidades de compilação v Contêm um grande número de funções que não estão nas mesmas unidades de compilação que seus chamadores

v Não execute um grande número de operações de entrada e saída

A otimização interprocessual está disponível apenas para programas ILE e programas de serviço que atendam às seguintes condições:

v Você criou os módulos vinculados ao programa ou programa de serviço especificamente para V4R4M0 ou versões posteriores.

v Você compilou os módulos vinculados ao programa ou programa de serviço com um nível de otimização de 20 (\*BASIC) ou superior.

v Os módulos vinculados ao programa ou programa de serviço têm dados IL associados a eles. Use a opção create module MODCRTOPT(\*KEEPILDTA) para manter os dados de linguagem intermediária (IL) com o módulo.

**Nota:** Devido aos requisitos de otimização, você deve depurar totalmente seus programas antes de usar a análise interprocessual.

## Como otimizar seus programas com IPA

Para usar o IPA para otimizar seu programa ou objetos de programa de serviço, execute as seguintes etapas:

1. Certifique-se de compilar todos os módulos necessários para o programa ou programa de serviço com MODCRTOPT(\*KEEPILDTA) e com um nível de otimização de 20 ou mais (de preferência 40). Você pode usar o comando DSPMOD com o parâmetro DETAIL(\*BASIC) para verificar se um único módulo é compilado com as opções corretas. O campo **Dados de idioma** **intermediário** terá um valor de \*SIM se os dados de IL estiverem presentes. O  campo Nível de otimização indica o nível de **otimização** do módulo.
2. Especifique IPA(\*YES) no comando CRTPGM ou CRTSRVPGM . Quando a parte IPA da associação é executada, o sistema exibe mensagens de status para indicar o progresso do IPA.

Você pode definir ainda mais como o IPA otimiza seu programa usando o seguinte parâmetro:

v Especifique IPACTLFILE(*IPA-control-file*) para fornecer informações adicionais de subopção IPA. Consulte "Sintaxe do arquivo de controle IPA" para obter uma listagem das opções que você pode especificar no arquivo de controle.

Quando você especifica IPA(\*YES) no comando CRTPGM, você também não pode permitir atualizações para o programa (ou seja, você não pode especificar ALWUPD(\*YES)). Isso também é verdadeiro para o parâmetro ALWLIBUPD no comando CRTSRVPGM. Se especificado junto com IPA(\*YES), o parâmetro deve ser ALWLIBUPD(\*NO).

## Sintaxe do arquivo de controle IPA

O arquivo de controle IPA é um arquivo de fluxo que contém diretivas de processamento IPA adicionais. O arquivo de controle pode ser um membro de um arquivo e usa o QSYS. Convenção de nomenclatura LIB (por exemplo, /qsys.lib/mylib.lib/ xx.file/yy.mbr). O parâmetro IPACTLFILE identifica o nome do caminho desse arquivo.

O IPA emite uma mensagem de erro se as diretivas do arquivo de controle tiverem sintaxe que não é válida.

Você pode especificar as seguintes diretivas no arquivo de controle:

#### exits=name[,nome]

Especifica uma lista de funções, cada uma das quais sempre encerra o programa. Você pode otimizar as chamadas para essas funções (por exemplo, eliminando sequências de salvamento e restauração), porque as chamadas nunca retornam ao programa. Essas funções não devem chamar nenhuma outra parte do programa que tenha dados de IL associados a elas.

#### inline=atributo

Especifica como você deseja que o compilador identifique as funções que você deseja que ele processe embutidas. Você pode especificar os seguintes atributos para esta diretiva:

**auto** Especifica que o embutido deve determinar se uma função pode ser embutida com base nos valores de limite embutido e limite embutido. A diretiva noinline substitui o inlining automático. Esse é o padrão.

#### noauto

Especifica que o IPA deve considerar para incluir somente as funções que você especificou por nome com a diretiva embutida.

#### nome[,nome]

Especifica uma lista de funções que você deseja incorporar. As funções podem ou não estar embutidas.

#### name[,name] de name[,name]

Especifica uma lista de funções que são candidatas desejáveis para inlining, se uma determinada função ou lista de funções chamar as funções. As funções podem ou não estar embutidas.

#### inline-limit=num

Especifica o tamanho relativo máximo (em unidades de código abstrato) para o qual uma função pode crescer antes de parar de alinhar. As unidades de código abstratas são proporcionais em tamanho ao código executável na função. Valores maiores para esse número permitem que o compilador incorpore subprogramas maiores, mais chamadas de subprogramas ou ambos. Esta diretiva é aplicável somente quando inline=auto está ativado. O valor padrão é 8192.

#### inline-threshold=size

Especifica o tamanho máximo (em unidades de código abstrato ) de uma função que pode ser candidata à insinuação automática. Esta diretiva é aplicável somente quando inline=auto está ativado. O tamanho padrão é 1024.

#### isolado=nome[,nome]

Especifica uma lista de funções "isoladas". Funções isoladas são aquelas que não se referem diretamente (ou indiretamente por meio de outra função dentro de sua cadeia de chamadas) ou alteram variáveis globais que são acessíveis. para funções visíveis. O IPA pressupõe que as funções vinculadas a programas de serviço sejam isoladas.

#### lowfreq=nome[,nome]

Especifica nomes de funções que se espera que sejam chamadas com pouca frequência. Normalmente, são funções de tratamento de erros ou funções de rastreamento. O IPA pode tornar outras partes do programa mais rápidas, fazendo menos otimização para chamadas para essas funções.

#### missing=atributo

Especifica o comportamento interprocessual de funções ausentes. As funções ausentes são aquelas que não têm dados de IL associados a elas e que não são explicitamente nomeadas em uma diretiva desconhecida, segura, isolada ou pura. Essas diretivas especificam quanta otimização o IPA pode executar com segurança em chamadas para rotinas de biblioteca que não têm dados de IL associados a elas.

O IPA não tem visibilidade do código dentro dessas funções. Você deve garantir que todas as referências de usuário sejam resolvidas com bibliotecas de usuário ou bibliotecas de tempo de execução.

A configuração padrão para essa diretiva é desconhecida. *Unknown* instrui o IPA a fazer suposições pessimistas sobre os dados que podem ser usados e alterados por meio de uma chamada para tal função ausente e sobre as funções que pode ser chamado indiretamente através dele. Você pode especificar os seguintes atributos para esta diretiva:

#### desconhecido

Especifica que as funções ausentes são "desconhecidas". Consulte a descrição da diretiva desconhecida abaixo. Esse é o atributo padrão.

**safe** Especifica que as funções ausentes são "seguras". Consulte a descrição da diretiva de segurança, abaixo.

#### isolado

Especifica que as funções ausentes são "isoladas". Veja a descrição da diretiva isolada, acima.

**pure** Especifica que as funções ausentes são "puras". Veja a descrição da diretiva pura, abaixo.

**noinline=nome[,nome]**

Especifica uma lista de funções que o compilador não irá incorporar.

#### noinline=name[,name] de name[,name]

Especifica uma lista de funções que o compilador não irá incorporar, se as funções forem chamadas de uma determinada função ou lista de funções.

#### partição=pequena| | médio grande | inteiro não assinado

Especifica o tamanho de cada partição de programa que o IPA cria. O tamanho da partição é diretamente proporcional ao tempo necessário para vincular e à qualidade do código gerado. Quando o tamanho da partição é grande, o tempo necessário para vincular é maior, mas a qualidade do código gerado geralmente é melhor.

O padrão para esta diretiva é médio.

Para um grau mais fino de controle, você pode usar um valor inteiro não assinado para especificar o tamanho da partição. O inteiro está em unidades de código abstratas , e seu significado pode mudar entre as versões. Você só deve usar esse inteiro para esforços de ajuste de curtíssimo prazo ou para aquelas situações em que o número de partições deve permanecer constante.

#### pure=nome[,nome]

Especifica uma lista de funções puras. Estas são funções que são seguras e isoladas. Uma função pura não tem estado interno observável. Isso significa que o valor retornado para uma determinada chamada de uma função é independente de quaisquer chamadas anteriores ou futuras da função.

#### safe=nome[,nome]

Especifica uma lista de funções seguras.  Essas são funções que não chamam direta ou indiretamente qualquer função que tenha dados de IL associados a ela. Uma função segura pode se referir e alterar variáveis globais.

**desconhecido=nome[,nome]**

Especifica uma lista de funções desconhecidas. Estas são funções que não são seguras, isoladas ou puras.

## Notas de uso do IPA

v O uso de IPA pode aumentar o tempo de ligação. Dependendo do tamanho do aplicativo e da velocidade do processador, o tempo de ligação pode aumentar significativamente.

v IPA pode gerar objetos de programa e programa de serviço vinculados significativamente maiores do que a associação tradicional.

v Embora as otimizações interprocessuais do IPA possam melhorar significativamente o desempenho de um programa, elas também podem fazer com que os programas em funcionamento que contêm erros falhem.

v Como o IPA compilará funções embutidas, tome cuidado ao usar APIs que aceitam um deslocamento de quadro de pilha relativo (por exemplo, QMHRCVPM).

v Para compilar funções embutidas, o IPA usa seu próprio inliner em vez do inliner de back-end. Todos os parâmetros fornecidos para o liner de back-end, como usar a opção INLINE no comando compile, são ignorados. Os parâmetros para o inliner IPA são fornecidos no arquivo de controle IPA.

## Restrições e limitações do IPA

v Você não pode usar o UPDPGM ou UPDSRVPGM em um programa acoplado ou programa de serviço que o IPA otimizou.

v Você não pode depurar nenhum programa ou programa de serviço que o IPA tenha otimizado com os recursos normais de depuração de origem. Isso ocorre porque o IPA não mantém informações de depuração dentro dos dados IL e, de fato, joga fora qualquer informação de depuração quando gera as partições de saída. Como tal, o depurador de origem não manipula programas IPA ou programas de serviço.

v Há um limite de 10.000 partições de saída. Se você atingir esse limite, a associação falhará e o sistema

enviará uma mensagem. Se você atingir esse limite, execute o comando CRTPGM ou CRTSRVPGM novamente e especifique um tamanho de partição maior. Consulte a diretiva de partição em "Sintaxe do arquivo de controle IPA" na página 132.

v Existem certas limitações de IPA que podem ser aplicadas ao seu programa se esse programa contiver dados SQL. Se o compilador que você usa permite uma opção para manter os dados IL, então essas limitações não se aplicam. Se o compilador usado não permitir uma opção para manter os dados IL, você deverá executar as etapas listadas abaixo para usar o IPA em um programa que contenha dados SQL. Por exemplo, considere um programa C com instruções SQL incorporadas. Normalmente , você compilaria esse código-fonte com o comando CRTSQLCI; no entanto, esse comando não tem uma opção MODCRTOPT(\*KEEPILDTA).

Execute as etapas a seguir para criar um \*MODULE que contenha dados SQL incorporados e dados IL.

1. Compile um arquivo de origem SQL C com o comando CRTSQLCI. Especifique as opções do compilador OPTION(\*NOGEN) e TOSRCFILE(QTEMP/QSQLTEMP). Esta etapa pré-compila as instruções SQL e coloca os dados do pré-compilador SQL no espaço associado do arquivo de origem original . Ele também coloca a fonte C em um membro com o mesmo nome no arquivo físico de origem temporária QTEMP/QSQLTEMP.
2. Compile o arquivo de origem C no QTEMP/QSQLTEMP com a opção MODCRTOPT(\*KEEPILDTA) no comando do compilador. Essa ação cria um objeto SQL C \*MODULE e propaga os dados do pré-processador do espaço associado do arquivo de origem original para o objeto de módulo. Esse objeto \*MODULE também contém os dados IL. Neste ponto, você pode especificar o objeto \*MODULE no comando CRTPGM ou CRTSRVPGM com o parâmetro IPA (\*YES). v IPA não pode otimizar módulos que você compila na otimização nível 10 (\*NENHUM). O IPA requer informações dentro dos dados de IL que estão disponíveis apenas em níveis de otimização mais altos.

v IPA não pode otimizar módulos que não contêm dados IL. Devido a isso, o IPA pode otimizar apenas os módulos que você cria com compiladores que oferecem a opção MODCRTOPT(\*KEEPILDTA). Atualmente, isso inclui os compiladores C e C++.

v Para um programa, o módulo que contém o ponto de entrada do programa, que normalmente é a função principal, deve ter os atributos corretos, conforme observado acima, ou o IPA falhará. Para um programa de serviço, pelo menos um dos módulos que contêm funções exportadas deve ter os atributos corretos, conforme observado acima, ou o IPA falhará. É desejável que os outros módulos dentro do programa ou programa de serviço também tenham os atributos corretos, mas não é necessário. Quaisquer módulos sem os atributos corretos serão aceitos pelo IPA, mas não serão otimizados.

v IPA pode não ser capaz de otimizar corretamente os módulos compilados com a opção RTBND(\*LLP64) em

o comando Criar módulo C++ (CRTCPPMOD) ou Criar programa C++ vinculado (CRTBNDCPP). Se as funções virtuais não forem usadas no módulo , o IPA poderá otimizar o módulo. Se funções virtuais forem usadas, a opção MODCRTOPT(\*NOKEEPILDTA) deve ser especificada.

v IPA pode não ser capaz de otimizar corretamente módulos que contêm dados ou variáveis decimais de ponto flutuante. v IPA não pode otimizar corretamente módulos que contêm variáveis de armazenamento local de thread.

## Partições criadas pelo IPA

O programa final ou programa de serviço criado pelo IPA consiste em partições. O IPA cria um \*MODULE para cada partição. As partições têm duas finalidades:

v Eles melhoram a localidade de referência em um programa, concentrando o código relacionado na mesma região de armazenamento. v Eles reduzem os requisitos de memória durante a geração de código objeto para que partição.

Existem três tipos de partições:

v Uma partição de inicialização. Isso contém código de inicialização e dados. v A partição primária. Isso contém informações para o ponto de entrada primário para o programa. v Partições secundárias ou outras.

O IPA determina o número de cada tipo de partição das seguintes maneiras:

v A diretiva 'partition' dentro do arquivo de controle especificado pelo parâmetro IPACTLFILE. Essa diretiva indica o tamanho de cada partição.

v A conectividade dentro do gráfico de chamada do programa. Conectividade refere-se ao volume de chamadas entre funções em um programa.

v Resolução de conflitos entre opções do compilador especificadas para diferentes unidades de compilação. O IPA tenta resolver conflitos aplicando uma opção comum em todas as unidades de compilação. Se não puder, ele força as unidades de compilação para as quais os efeitos da opção original devem ser mantidos em partições separadas.

Um exemplo disso são as *Opções*  de  *Código Interno Licenciado* (LICOPTs). Se duas unidades de compilação tiverem LICOPTs conflitantes, o IPA não poderá mesclar funções dessas unidades de compilação na mesma partição de saída. Consulte "Mapa de partição " na página 164 para obter um exemplo da seção de listagem Mapa de partição. O IPA cria as partições em uma biblioteca temporária e vincula os \*MODULEs associados para criar o programa final ou o programa de serviço. O IPA cria os nomes de partição \*MODULE usando um prefixo aleatório (por exemplo, QD0068xxxx onde xxxx varia de 0000 a 9999).

Devido a isso, alguns dos campos dentro DSPPGM ou DSPSRVPGM podem não ser como esperado. O

'Módulo de procedimento de entrada no programa' mostra o nome da partição \*MODULE e não o original

\*Nome do módulo. O campo 'Biblioteca' desse módulo mostra o nome da biblioteca temporária em vez do nome da biblioteca original. Além disso, os nomes dos módulos vinculados ao programa ou programa de serviço serão os nomes de partição gerados. Para qualquer programa ou programa de serviço que tenha sido otimizado pelo IPA, o campo 'Atributo do programa' exibido pelo DSPPGM ou DSPSRVPGM será IPA, assim como o campo de atributo de todos os módulos acoplados para esse programa ou programa de serviço.

**Nota:** Quando o IPA está fazendo particionamento, o IPA pode prefixar a função ou o nome dos dados com @nnn@ ou

XXXX@nnn@, onde XXXX é o nome da partição e onde nnn é o número do arquivo de origem. Isso garante que os nomes de função estática e os nomes de dados estáticos permaneçam exclusivos.

# Otimização avançada de argumentos

A otimização avançada de argumentos é uma otimização entre módulos usada para melhorar o desempenho de programas que contêm chamadas de procedimento executadas com frequência, incluindo aplicativos C++ que fazem principalmente chamadas de método não virtuais . O desempenho aprimorado do tempo de execução é alcançado permitindo que o conversor e o fichário usem os mecanismos mais eficientes para passar parâmetros e retornar resultados entre procedimentos que são chamado dentro de um programa ou programa de serviço.

**Como usar a otimização avançada de argumentos**

| O parâmetro Argument optimization (ARGOPT), com \*SIM e \*NO como valores possíveis, está disponível em

| os comandos CRTPGM e CRTSRVPGM para oferecer suporte à otimização avançada de argumentos. Especificando | ARGOPT(\*YES) faz com que o programa ou programa de serviço seja criado com argumento avançado | otimização. O padrão é \*NO.

**Considerações e restrições ao usar a otimização avançada de argumentos**

Quando ARGOPT(\*YES) é especificado durante a criação do programa, a otimização avançada de argumentos é aplicada. Em geral, isso melhorará o desempenho da maioria das chamadas de procedimento dentro do programa. No entanto, você precisa considerar os seguintes itens antes de decidir usar a otimização avançada de argumentos:

v Interação com a otimização de argumentos baseada em pragma:

A otimização de argumento habilitada com ARGOPT(\*YES) e a otimização de argumento habilitada pela diretiva argopt #pragma, que é suportada pelos compiladores C e C++, são redundantes e complementares.

Se você já tem #pragma argopt em seu código, deixe-o lá, e também use ARGOPT(\*YES). Você pode remover os #pragma redundantes mais tarde ou mantê-los lá.

Se você não tiver #pragma argopt em seu código, o uso de ARGOPT(\*YES) geralmente ajudará. Se você chamar procedimentos por meio de um ponteiro de função, convém pensar em usar #pragma argopt para esses casos, pois a otimização avançada de argumentos não otimiza as chamadas através de um ponteiro de função. Chamadas de função virtual em C++ são exemplos de chamadas de ponteiro de função.

| Para obter mais informações sobre a #pragma diretiva argopt, consulte ILE C/C++ Compiler Reference .

No entanto, ao contrário da otimização de argumento baseada em pragma, que requer a inserção manual de diretivas de #pragma no código-fonte , a otimização avançada de argumentos não requer alterações no código-fonte e é aplicado automaticamente. Além disso, a otimização avançada de argumentos pode ser aplicada a programas criados em qualquer linguagem, enquanto a solução baseada em pragma é apenas para C e C++.

Embora a diretiva argopt #pragma possa ser aplicada a ponteiros de função, a otimização avançada de argumentos não otimiza automaticamente as chamadas de função virtual e as chamadas por meio de um ponteiro de função. Portanto, para otimizar chamadas indiretas, o pragma argopt é útil quando usado dessa maneira complementar com otimização avançada de argumentos. v ponteiros de 16 bytes:

Os parâmetros de ponteiro de espaço de 16 bytes são os que mais se beneficiam da otimização de argumentos. Os ponteiros de espaço apontam para tipos de objeto de dados, como caracteres, números, classes e estruturas de dados. Exemplos de ponteiros de espaço em C e C++ incluem char\* e int\*. No entanto, os parâmetros declarados com outros tipos de ponteiros de 16 bytes que são exclusivos do IBM i, como ponteiros que apontam para objetos do sistema, não são otimizados pela otimização de argumentos. Parâmetros de ponteiro abertos, como ponteiros void\* em C e C++, que são baseados em tipos incompletos, também não são otimizados. v DTAMDL(\*LLP64):

Os aplicativos C e C++ que consistem em módulos criados com DTAMDL(\*LLP64) se beneficiam menos da otimização de argumentos do que aqueles criados com o DTAMDL(\* P128) padrão. No primeiro caso, os ponteiros para os dados têm 8 bytes de comprimento, e estes são sempre passados entre os procedimentos usando os mecanismos mais eficientes. Neste último caso, os ponteiros para os dados têm 16 bytes, que são os principais candidatos para otimização de argumentos. v Versão de destino:

Os programas criados com ARGOPT(\*YES) também devem ser criados com a versão de destino V6R1M0 ou posterior.

Para aproveitar ao máximo a otimização avançada de argumentos, os módulos vinculados a programas criados com ARGOPT(\*YES) devem ser criados com a versão de destino V6R1M0 ou posterior, como chamadas ou chamadas originadas de funções definidas em módulos criados antes de V6R1M0 são ignoradas pela otimização avançada de argumentos.

v Maior tempo de criação do programa:

Quando ARGOPT(\*YES) é especificado durante a criação do programa, uma análise adicional é executada em todos os módulos do programa . Para programas que consistem em centenas ou milhares de módulos, o tempo de criação pode ser significativamente maior.

Da mesma forma, quando você usa os comandos Programa de Atualização (UPDPGM) ou Programa de Serviço de Atualização (UPDSRVPGM) para atualizar um programa que é criado com ARGOPT(\*YES), a quantidade de tempo para concluir a atualização pode ser mais longo devido à análise extra que pode ser necessária para garantir que todas as chamadas entre módulos sejam atualizadas. Se não houver alterações nas interfaces de procedimento, esse tempo extra geralmente é curto. v Interação com convenções de chamada especiais:

A otimização avançada de argumentos não é aplicável a chamadas dinâmicas de programas. Além disso, as funções C e C++ definidas com a palavra-chave \_System não são candidatas à otimização avançada de argumentos. v Interação com a criação de perfil do programa:

Você pode usar a otimização avançada de argumentos e a criação de perfil de programa ao mesmo tempo. v Interação com a análise interprocessual (IPA):

A análise entre módulos e as otimizações realizadas pelo IPA são redundantes com a otimização avançada de argumentos. Portanto, quando você usa IPA, não é necessário usar a otimização avançada de argumentos.

**Opções** de  **código interno** licenciado

*Opções de Código Interno Licenciado (LICOPTs)*  são opções do compilador que são passadas para o Código Interno Licenciado para afetar a forma como o código é gerado ou empacotado. Essas opções afetam o código que é gerado para um módulo. Você pode usar algumas das opções para ajustar a otimização do seu código. Algumas das opções ajudam na depuração. Esta seção discute as opções de Código Interno Licenciado.

**Opções definidas no momento**

As opções de Código Interno Licenciado que estão definidas atualmente são:

#### [Não] CallTracingAtHighOpt

Use essa opção para solicitar que os traps de chamada e retorno sejam inseridos no prólogo e no epílogo do procedimento, respectivamente, de procedimentos que exigem uma pilha, mesmo no nível de otimização 40. A vantagem de inserir interceptações de chamada e retorno é a capacidade de usar o rastreamento de trabalho, enquanto a desvantagem é potencialmente pior desempenho de tempo de execução . Em versões anteriores à 6.1, essa opção está desativada por padrão, o que significa que nenhum trap de chamada e retorno é inserido em nenhum procedimento no nível de otimização 40. A partir da versão 6.1, essa opção é ignorada e os traps de chamada e retorno são inseridos incondicionalmente em procedimentos que exigem uma pilha, mesmo no nível de otimização 40.

#### [Não] Compacto

Use essa opção para reduzir o tamanho do código sempre que possível, às custas da velocidade de execução. Isso é feito inibindo otimizações que replicam ou expandem o código embutido. Essa opção está desativada por padrão.

#### CodeGenTarget=

A opção CodeGenTarget especifica o modelo de destino de criação para um objeto de programa ou módulo. O modelo de destino de criação indica os recursos de hardware que o código gerado para esse objeto pode usar. Consulte a tabela a seguir para obter os valores possíveis deste LICOPT.

|  |  |
| --- | --- |
| **Valor** | **Significado** |
| CodeGenTarget=Atual | O tradutor otimizador pode usar todos os recursos disponíveis na máquina atual. |
| CodeGenTarget=Comum | O tradutor otimizador pode usar qualquer recurso disponível em todos os sistemas suportados pela versão de destino. |
| CodeGenTarget=Legado | O conversor de otimização não pode usar nenhum recurso que não esteja disponível antes da versão 6.1. |
| CodeGenTarget=Potência6 | O conversor otimizador pode usar todos os recursos disponíveis no nível POWER6 da arquitetura PowerPC® AS. |
| CodeGenTarget=Potência7 | O conversor otimizador pode usar todos os recursos disponíveis no nível POWER7®  da arquitetura PowerPC AS. |

|

|

Para obter mais informações sobre essa opção, consulte "O CodeGenTarget LICOPT" na página 146.

#### [Não] CriarSuperblocos

Essa opção controla a formação de superblocos, que são grandes blocos básicos estendidos sem entrada de fluxo de controle, exceto no cabeçalho do superbloco. Ele também controla certas otimizações realizadas em superblocos, como desenrolamento de traços e descascamento de traços. A formação e otimizações de superblocos podem causar grande quantidade de duplicação de código; este LICOPT pode ser usado para desativar essas otimizações. Este LICOPT só é eficaz quando os dados de perfil são aplicados. Essa opção está ativada por padrão.

#### [Não] DetectConvertTo8BytePointerError

Essa opção é ignorada em sistemas que executam a versão 6.1 ou posterior. Cada conversão de um ponteiro de 16 bytes para um ponteiro de 8 bytes sinaliza uma exceção MCH0609 se o ponteiro de 16 bytes não contiver um endereço teraspace e não contiver um valor de ponteiro nulo.

#### [Não] AtivandoInlining

Essa opção controla o procedimento embutido pelo tradutor de otimização. Entrada de procedimento significa que uma chamada para um procedimento é substituída por uma cópia embutida do código de procedimento. Essa opção está ativada por padrão. **[Não]** **FoldFloat**

Especifica que o sistema pode avaliar expressões constantes de ponto flutuante em tempo de compilação. Este LICOPT substitui a opção de criação do módulo 'Fold float constants'. Quando este LICOPT não é especificado, a opção de criação do módulo é honrada.

#### LoopUnrolling=<opção>

Use a opção LoopUnrolling para controlar a quantidade de desenrolamento de loop realizado pelo conversor de otimização. Os valores válidos são 0 para desabilitar a desrolagem de loops, 1 para desenrolar loops pequenos com ênfase na redução da duplicação de código e 2 para desenrolar loops agressivamente. O uso da opção 2 pode aumentar substancialmente o tamanho do código gerado. O valor padrão é 1.

#### [Não] Maf

Permitir a geração de instruções de multiplicação-adição de ponto flutuante. Estas instruções combinam uma operação de multiplicação e uma operação de adição sem uma operação de arredondamento intermediária. O desempenho de execução é melhorado, mas os resultados computacionais podem ser afetados. Este LICOPT substitui a opção de criação do módulo 'Usar adicionar multiplicar'. Quando este LICOPT não é especificado, a opção de criação do módulo é honrada.

#### [Não] MinimizeTeraspaceFalseEAOs

| A verificação efetiva de estouro de endereço (EAO) é feita como parte das operações aritméticas de endereço para

| Ponteiros de 16 bytes. O mesmo código gerado deve lidar com teraspace e armazenamento de nível único (SLS)

| endereços, portanto, usos válidos de teraspace podem incorrer em EAOs falsos se o código foi gerado para um processador

| antes de POWER6. Consulte "Adaptive Code Generation" na página 143 para obter mais informações. Estes EAO

| as condições não indicam um problema, mas manuseá-las adiciona uma sobrecarga significativa de processamento. O

| MinimizeTeraspaceFalseEAOs LICOPT causa diferenças nas sequências de instruções de hardware

| gerado para o programa. Diferentes sequências de instruções aritméticas de endereços são geradas que são

| um pouco mais lento no caso usual, mas eliminar a maioria das ocorrências EAO. Um exemplo de quando isso | LICOPT deve ser usado é quando a maioria da aritmética de endereços realizada em um módulo calcula teraspace | endereçar os resultados a uma área de 16 MB de valor inferior. Essa opção está desativada por padrão.

#### [Não] OrderedPtrComp (em inglês)

Use essa opção para comparar ponteiros como valores inteiros não assinados e para sempre produzir um resultado ordenado (igual, menor ou maior que). Quando você usa essa opção, os ponteiros que se referem a espaços diferentes não serão comparados não ordenados. Essa opção está desativada por padrão.

#### [Não] PreverRamificaçõesEmAusênciaDePerfil

Quando os dados de perfil não forem fornecidos, use essa opção para executar a previsão de ramificação estática para orientar otimizações de código. Se os dados de perfil forem fornecidos, os dados de perfil serão usados para prever probabilidades de ramificação, independentemente dessa opção. Essa opção está desativada por padrão.

#### [Não] PtrDisjunta

Essa opção permite um refinamento agressivo de alias baseado em tipo que permite que o conversor otimizador elimine um conjunto maior de cargas redundantes, o que pode melhorar o desempenho do tempo de execução. Um aplicativo pode usar com segurança essa opção se o conteúdo de um ponteiro não for acessado por meio de um tipo que não seja ponteiro. A expressão a seguir em C demonstra uma maneira insegura de acessar o valor de um ponteiro:

void\* spp;

... = ((long long\*) &spp) [1]; // Acessar 8 bytes de ponteiro de 16 bytes de ordem baixa

Padrão: NoPtrDisjoint

#### [Não] ReassocForIndexedLdSt

Essa opção instrui o tradutor otimizador a reassociar expressões de endereçamento que alimentam instruções de carregamento ou armazenamento para que elas atendam aos requisitos de conversão em carga indexada ou instruções de armazenamento. Na maioria dos casos, isso significa procurar instruções de carga ou armazenamento com um deslocamento diferente de zero que poderiam ser transformadas em cargas indexadas ou instruções de armazenamento se o deslocamento foi zero. Nesse caso, a reassociação gera uma adição explícita do deslocamento na expressão de endereço e o deslocamento nas instruções de carregamento ou armazenamento é zero.

O padrão deste Código Interno Licenciado é ReassocForIndexedLdSt.

#### [Não] TailCallOptimizations

Essa opção instrui o tradutor de otimização, quando compila no nível de otimização 40, a executar otimizações de chamada de cauda que tentam reduzir o número de quadros de pilha que são mantido na pilha de chamadas de tempo de execução. A vantagem disso é que, em alguns casos, menos quadros de pilha podem ser necessários, o que pode melhorar a localidade de referência e, em raras circunstâncias, reduzir a possibilidade de estouro de pilha de tempo de execução. A desvantagem é que, se um programa falhar, menos pistas são deixadas na pilha de chamadas que podem ser usadas para depuração. Recomenda-se que este LICOPT permaneça ativado.

Uma chamada de *cauda* é uma chamada no procedimento que é feita imediatamente antes de retornar . Nesses casos, o tradutor otimizador tenta executar otimizações de chamada de cauda que eliminam a necessidade de o chamador alocar um quadro de pilha e deixar a pilha inalterada . Normalmente, um quadro de pilha é criado pelo chamador para permitir que o destinatário salve e restaure o endereço de retorno. Com essas otimizações, a chamada de cauda é alterada para uma operação de ramificação simples que não calcula um novo endereço de retorno. O endereço de retorno original é deixado intacto e, portanto, aponta de volta para o local de retorno do chamador. As otimizações eliminam o quadro de pilha intermediário entre o chamador e o chamado, e o destinatário retorna ao chamador do chamador.

Por exemplo, se a Função A chamar a Função B e a Função B fizer uma chamada de cauda para a Função C, com otimizações de chamada de cauda habilitadas, a Função B não alocará um quadro de pilha e, em vez disso, ramificações para a Função C. Quando a Função C estiver concluída, ela retornará diretamente para a Função A após a chamada para a Função B. Sem otimizações de chamada de cauda, a Função C retorna à Função B, que retorna imediatamente à Função A.

Para procedimentos chamados com frequência, as otimizações de chamada de cauda podem melhorar o desempenho , eliminando as operações necessárias para criar e destruir um quadro de pilha desnecessário. Se você especificar o valor LICOPT NoTailCallOptimizations, essas otimizações não serão tentadas.

O padrão deste LICOPT é TailCallOptimizations.

#### TargetProcessorModel=<opção>

| A opção TargetProcessorModel instrui o conversor a executar otimizações para o especificado

| modelo do processador. Os programas criados com esta opção são executados em todos os modelos de hardware suportados, mas

| geralmente são executados mais rapidamente no modelo de processador especificado. Os valores válidos são 0 para processadores Star, 2 para

| Processadores POWER4, 3 para processadores POWER5, 4 para processadores POWER6 e 5 para POWER7

| Processadores. O valor padrão depende da liberação de destino associada ao objeto de programa. Para | IBM i V5R2 e IBM i V5R3, o valor padrão é 2. Para o IBM i V5R4, o valor padrão é 3. Para IBM i | 6.1, o valor padrão é 4. Para o IBM i 7.1, o valor padrão é 5.

| O TargetProcessorModel LICOPT é um dos vários fatores que determinam qual modelo de processador | deve ser direcionado ao criar um módulo , alterar um módulo ou módulo acoplado ou recriar um | módulo ou módulo acoplado. As regras a seguir se aplicam a módulos e módulos acoplados.

| v Quando um módulo é criado ou alterado e o TargetProcessorModel LICOPT é especificado, o | O código gerado para o módulo é ajustado para obter o melhor desempenho no modelo de processador especificado, | independentemente de o CodeGenTarget LICOPT também ser especificado.

| v Quando um módulo é criado ou alterado e o TargetProcessorModel LICOPT não é especificado, mas | o CodeGenTarget LICOPT é especificado como POWER6 ou POWER7, o código gerado para o | O módulo é ajustado para obter o melhor desempenho no modelo de processador especificado.

| v Quando um módulo é criado ou alterado e o TargetProcessorModel LICOPT não é especificado, mas | o CodeGenTarget LICOPT é especificado como Current, o código gerado para o módulo é ajustado | para melhor desempenho no processador em uso pela partição na qual o módulo reside.

| v Quando um módulo é criado ou alterado e o TargetProcessorModel LICOPT não é especificado, mas

| o CodeGenTarget LICOPT é especificado como Common, então o módulo é ajustado para melhor | desempenho em um modelo de processador padrão para a versão. O modelo de processador padrão é POWER7 | para a versão 7.1, POWER6 para a versão 6.1 e POWER5 para versões anteriores suportadas.

| v Quando um módulo é criado ou alterado e o TargetProcessorModel LICOPT não é especificado, mas

| o CodeGenTarget LICOPT é especificado como Legacy, então o módulo é ajustado para melhor | desempenho no modelo de processador POWER5.

| v Quando um módulo é recriado, o módulo é ajustado para obter o melhor desempenho no processador em uso pelo | a partição na qual o módulo reside, independentemente dos valores de qualquer TargetProcessorModel | e CodeGenTarget LICOPTs especificados quando o módulo foi criado ou alterado pela última vez.

Observe que para a maioria dessas opções há uma variação positiva e uma negativa, a negativa começando com o prefixo ' não'. A variante negativa significa que a opção não deve ser aplicada. Sempre haverá duas variantes como essa para as opções booleanas, a fim de permitir que um usuário desative explicitamente uma opção, bem como ative-a. A capacidade de fazer isso é necessária para desativar uma opção para a qual a opção padrão está ativada. O padrão para qualquer opção pode mudar de versão para versão.

**Aplicação**

Você pode especificar opções de Código Interno Licenciado (LICOPTs) ao criar módulos e pode modificar as opções de um objeto existente com o Módulo de Alteração (CHGMOD), Alterar Programa ( CHGPGM) e comandos do Programa de Serviço de Alteração (CHGSRVPGM). Com esses comandos, você pode substituir toda a cadeia de caracteres LICOPT ou adicionar LICOPTs a uma cadeia de caracteres existente. Adicionar LICOPTs a uma cadeia de caracteres existente é útil quando os objetos que estão sendo alterados têm LICOPTs diferentes e você não deseja perder os LICOPTs existentes.

Um exemplo de aplicação de opções de Código Interno Licenciado a um módulo é:

> MÓDULO CHGMOD(TESTE) LICOPT('maf')

Quando usado no CHGPGM ou CHGSRVPGM, o sistema aplica as opções de Código Interno Licenciado especificadas a todos os módulos contidos no objeto de programa ILE. Um exemplo de aplicação de opções de Código Interno Licenciado a um objeto de programa ILE é:

> CHGPGM PGM(TESTE) LICOPT('nomaf')

Um exemplo de aplicação de opções de Código Interno Licenciado a um programa de serviço é:

> CHGSRVPGM SRVPGM(TEST) LICOPT('maf')

Para adicionar um LICOPT a um objeto existente, use a palavra-chave \*ADD no parâmetro LICOPT. Este exemplo preserva os LICOPTs existentes do objeto e adiciona um novo: LICOPT ('maf', \*ADD).

**Restrições**

Existem várias restrições sobre os tipos de programas e módulos aos quais você pode aplicar as opções de Código Interno Licenciado.

v Você não pode aplicar as opções de Código Interno Licenciado a programas OPM.

v O módulo ou o programa ILE ou o objeto de programa de serviço deve ter sido originalmente criado para a versão V4R5M0 ou posterior.

v Você não pode aplicar as opções de Código Interno Licenciado a módulos vinculados pré-V4R5 dentro de um programa ou programa de serviço V4R5 ou posterior. Isso não afeta outros módulos vinculados dentro do programa que podem ter LICOPTs aplicados.

**Sintaxe**

Nos comandos CHGMOD, CHGPGM e CHGSRVPGM, o caso do valor do parâmetro LICOPT não é significativo. Por exemplo, as duas invocações de comando a seguir teriam o mesmo efeito:

> CHGMOD MODULE(TEST) LICOPT('nomaf')

> MÓDULO CHGMOD(TESTE) LICOPT('NoMaf')

Ao especificar várias opções de Código Interno Licenciado juntas, você deve separar as opções por vírgulas. Além disso, o sistema ignora todos os espaços que precedem ou que seguem qualquer opção. Aqui estão alguns exemplos:

> MÓDULO CHGMOD (TESTE) LICOPT('Maf, NoFoldFloat')

> MÓDULO CHGMOD (TESTE) LICOPT('Maf, NoFoldFloat')

> MÓDULO CHGMOD (TESTE) LICOPT(' Maf, NoFoldFloat ')

Para opções booleanas, o sistema não permite a especificação das duas variantes opostas ao mesmo tempo. Por exemplo, o sistema não permitirá o seguinte comando:

> CHGMOD MODULE(TEST) LICOPT('Maf,NoMaf') <- NÃO PERMITIDO!

No entanto, você pode especificar a mesma opção mais de uma vez. Por exemplo, isso é válido:

> MÓDULO CHGMOD (TESTE) LICOPT('Maf, NoFoldFloat, Maf')

**Compatibilidade de** versão

O sistema não permitirá que os usuários movam um módulo, programa ou programa de serviço que tenha tido opções de Código Interno Licenciado aplicadas a qualquer versão anterior à V4R5M0. Na verdade, o sistema impede que o usuário especifique uma versão de destino anterior ao tentar salvar o objeto na mídia ou em um arquivo de salvamento.

Às vezes, o IBM i define novas opções de Código Interno Licenciado em novas liberações (ou dentro de uma determinada liberação por meio de um PTF). Você pode usar as novas opções em sistemas que tenham a primeira versão que as ofereça suporte ou em qualquer versão posterior. Você pode mover qualquer módulo, programa ou programa de serviço que tenha novas opções aplicadas para uma versão que não ofereça suporte às opções. O sistema ignora e não aplica mais as opções de Código Interno Licenciado sem suporte para objetos convertidos se o parâmetro LICOPT de um comando não especificar as opções. Os valores LICOPT sem suporte serão ignorados se a recriação ocorrer quando você usar LICOPT(\*SAME) nos comandos CHGMOD, CHGPGM ou CHGSRVPGM. Eles também são ignorados quando a recriação ocorre quando o sistema converte automaticamente o objeto. Por outro lado, qualquer tentativa de especificar opções sem suporte no parâmetro LICOPT dos comandos CHGMOD, CHGPGM ou CHGSRVPGM falhará.

**Exibindo opções** de  **código interno** licenciado do módulo e  **do** **programa ILE**

Os comandos DSPMOD, DSPPGM e DSPSRVPGM exibem as opções de Código Interno Licenciado que foram aplicadas. DSPMOD os exibe na seção Informações do módulo. Por exemplo:

Opções de Código Interno Licenciado . . . . . . . . . . : maf

DSPPGM e DSPSRVPGM exibem as opções de Código Interno Licenciado que são aplicadas a cada módulo individual dentro do programa na seção Atributos do módulo para cada módulo.

Ao especificar a mesma opção de Código Interno Licenciado mais de uma vez, todas as ocorrências dessa opção, exceto a última, aparecem precedidas por um símbolo '+' . Por exemplo, suponha que o comando usado para aplicar as opções de Código Interno Licenciado a um objeto de módulo seja o indicado abaixo:

> MÓDULO CHGMOD(TESTE) LICOPT('maf, maf, Maf')

Em seguida, o DSPMOD mostrará isso:

Opções de Código Interno Licenciado . . . . . . . . . . : +maf,+maf,Maf

O '+' significa que o usuário especificou ocorrências redundantes da mesma opção.

Se alguma opção de Código Interno Licenciado aparecer precedida por um símbolo '\*', elas não se aplicam mais a um Módulo ou Programa ILE. Isso ocorre porque o sistema que executou a última recriação do objeto não os suportou. Para obter mais informações, consulte a seção "Compatibilidade de versão". Por exemplo, suponha que a nova opção foi originalmente aplicada em um sistema de versão N+1 usando o seguinte comando:

> CHGMOD MODULE(TEST) LICOPT('NewOption')

O Module é levado de volta para um sistema N de versão que não oferece suporte a essa opção e, em seguida, o objeto Module é recriado usando:

> CHGMOD MODULE(TEST) FRCCRT(\*YES) LICOPT(\*SAME)

As opções de Código Interno Licenciado mostradas no DSPMOD terão esta aparência:

Opções de Código Interno Licenciado . . . . . . . . . . : \*NewOption

O '\*' significa que a opção não se aplica mais ao Módulo.

**Geração** **de código** adaptável

Geralmente, você não precisa entender os detalhes da arquitetura de hardware subjacente para que a arquitetura possa mudar suavemente ao longo do tempo. As alterações de arquitetura vão desde a adição de uma única instrução de processador até a alteração de todo o conjunto de instruções do processador. Para garantir que os programas continuem a ser executados corretamente quando o sistema operacional se move entre plataformas que têm diferentes níveis de hardware subjacente, uma interface de máquina abstrata (MI) que representa programas em um formato independente de hardware é usado.

O tradutor otimizador é responsável por gerar instruções de hardware a partir da representação MI. Como o tradutor otimizador é um componente do sistema operacional, há uma única versão do tradutor otimizador para cada versão. No entanto, uma determinada versão pode ser suportada em vários modelos de sistema diferentes, cada um dos quais pode ter hardware de processador ligeiramente diferente dos outros.

Em versões anteriores à 6.1, o tradutor de otimização para uma determinada versão foi projetado para gerar apenas instruções que seriam executadas em todos os modelos de sistema suportados por essa versão . A vantagem dessa política é que um programa compilado para uma versão específica pode ser executado inalterado em qualquer sistema que execute a mesma versão. Isso facilita a criação e a distribuição de software para cada versão. No entanto, novos recursos importantes do processador, que muitas vezes fornecem vantagens significativas de desempenho , não podem ser usados até que estejam presentes em todos os sistemas suportados pelo atual soltar. Pode ocorrer um intervalo de vários anos entre a disponibilidade de um recurso de processador e seu uso em seus programas.

A partir da versão 6.1, você pode aproveitar todos os recursos do processador em seus sistemas, independentemente de esses recursos estarem presentes em outros modelos de sistema suportados pela mesma versão. Além disso, os programas podem ser movidos de um modelo de sistema para outro e continuar a ser executados corretamente, mesmo que a nova máquina não tenha todos os recursos de processador disponíveis. no original. A tecnologia usada para conseguir isso é chamada de geração de código adaptável. A geração de código adaptável (ACG) pode funcionar sem a intervenção do usuário para a maioria dos cenários. No entanto, se você criar e distribuir software para ser executado em uma variedade de modelos de sistema, convém exercer algum controle sobre quais recursos do processador são usados por geração de código adaptável.

**Conceitos** ACG

Para entender como a geração de código adaptável (ACG) funciona, é útil entender os seguintes conceitos.

Um recurso de hardware é um recurso que foi incluído na família de processadores suportados pelo IBM i. Por exemplo, um novo recurso disponível em alguns processadores suportados pela 6.1 é uma nova unidade de ponto flutuante decimal de hardware. Um processador com esta unidade é considerado pelo ACG como tendo o recurso de ponto flutuante decimal, enquanto um processador sem ele não tem o recurso de ponto flutuante decimal . A agregação de todos os recursos adicionados presentes em um processador é chamada de conjunto de recursos desse processador .

Um *modelo de destino* é uma abstração de todos os processadores que têm o mesmo conjunto de recursos. Um exemplo de um modelo de destino seriam todos os processadores que estão em conformidade com o nível POWER6 da arquitetura PowerPC AS.

Um módulo ou objeto de programa também tem um conjunto de recursos que identifica os recursos exigidos pelo objeto para que o objeto seja executado corretamente sem alterações. Um objeto é compatível com um modelo de destino se todos os recursos no conjunto de recursos do objeto também estiverem presentes no conjunto de recursos do modelo de destino.

*A geração* de código refere-se ao processo de criação de instruções de hardware para o módulo e os objetos de programa. A geração de código é realizada pelo tradutor otimizador.

Um módulo ou objeto de programa pode ser movido de um sistema para outro. O sistema no qual um objeto reside é chamado de *máquina atual*.

**Operação Normal**

Quando você compila um objeto de módulo usando um comando como Criar módulo C (CRTCMOD), o conversor de otimização detecta automaticamente quais recursos do processador estão disponíveis em seu sistema. As instruções de hardware geradas para o objeto de módulo aproveitam todos os recursos opcionais do processador que possam ser úteis. O conversor de otimização armazena o conjunto de recursos usado no objeto module como parte do objeto.

Quando você cria um objeto de programa usando um comando como Criar programa (CRTPGM) ou Criar programa de serviço (CRTSRVPGM), o fichário determina o conjunto de recursos para o objeto de programa e o armazena como parte do objeto de programa. Um recurso é incluído no conjunto de recursos do programa se estiver presente em qualquer um dos conjuntos de recursos dos módulos do programa.

A primeira vez que o objeto de programa é ativado em seu sistema , o sistema verifica se o objeto de programa é compatível com o modelo de destino associado ao seu sistema; ou seja, ele garante que seu programa não use nenhum recurso que não esteja disponível em seu sistema. Como você compilou o objeto de programa neste sistema, o objeto de programa sempre passa nessa verificação de compatibilidade e o programa é executado corretamente.

Suponha que você queira migrar esse objeto de programa para outro sistema que usa a mesma versão, mas tem um modelo de destino diferente. A primeira vez que o programa é ativado no sistema para o qual é movido, o sistema executa a verificação de compatibilidade em relação ao modelo de destino deste sistema. Se o programa for compatível com o sistema, o programa será executado corretamente. No entanto, se o programa requer qualquer recurso de processador que não é suportado pelo sistema para o qual ele foi movido, em seguida, o sistema chama automaticamente o otimizando o tradutor para converter o programa para ser compatível. O tradutor detecta quais recursos estão disponíveis no novo sistema e aproveita todos os que são aplicáveis, assim como fazia quando o módulo original se opõe foram criados. O programa convertido é então ativado conforme solicitado.

**Opções** de restauração

Para alterar o comportamento da geração de código adaptável quando os objetos de módulo e programa são restaurados em seu sistema , defina o valor do sistema Forçar Conversão na Restauração (QFRCCVNRST) e Forçar Conversão de Objeto ( FRCOBJCVN) parâmetro de comando. O parâmetro FRCOBJCVN está presente nos comandos Restore (RST), Restore Object (RSTOBJ) e Restore Library (RSTLIB ).

**Valor** do sistema QFRCCVNRST

O valor do sistema Forçar conversão na restauração (QFRCCVNRST) tem os seguintes valores possíveis.

|  |  |
| --- | --- |
| **Valor** | **Significado** |
| 0 | Não converta nada. |
| 1 | Objetos com erros de validação são convertidos. (Este é o padrão.) |
| 2 | Objetos que exigem conversão na versão atual do sistema operacional ou na máquina atual e objetos com erros de validação são convertidos. |
| 3 | Objetos suspeitos de terem sido violados, objetos que contêm erros de validação e objetos que exigem conversão na versão atual do sistema operacional ou em a máquina atual é convertida. |
| 4 | Os objetos que contêm dados de criação suficientes para serem convertidos e não têm assinaturas digitais válidas são convertidos. |
| 5 | Os objetos que contêm dados de criação suficientes são convertidos. |
| 6 | Todos os objetos que não têm assinaturas digitais válidas são convertidos. |
| 7 | Todos os objetos são convertidos. |

Ao definir QFRCCVNRST como 2 ou 3, você pode garantir que qualquer programa incompatível e objetos de módulo sejam convertidos imediatamente durante o processamento do comando Restore (RST, RSTOBJ, RSTLIB), em vez de no primeira ativação. Isso pode ser preferível em alguns casos, porque o tempo para converter um programa pode ser longo, particularmente para programas grandes e programas que são compilados em alta otimização níveis. Se você geralmente restaura programas em um sistema com o qual eles podem ser incompatíveis, considere alterar esse valor do sistema.

**Parâmetro** FRCOBJCVN

O parâmetro Force Object Conversion (FRCOBJCVN) nos comandos Restore (RST), Restore Object ( RSTOBJ) e Restore Library (RSTLIB) pode ser usado para exercer controle sobre a geração de código adaptável. A tabela a seguir mostra os valores possíveis para esses comandos. O comando RSTOBJ é usado como exemplo.

|  |  |
| --- | --- |
| **Valor** | **Significado** |
| RSTOBJ FRCOBJCVN(\*SYSVAL) | Os objetos são convertidos com base no valor do valor do sistema Force Conversion on Restore (QFRCCVNRST). (Este é o padrão.) |
| RSTOBJ FRCOBJCVN(\*NÃO) | Os objetos não são convertidos durante a operação de restauração. |
| RSTOBJ FRCOBJCVN(\*SIM \*RQD) | Os objetos são convertidos durante a operação de restauração somente se exigirem que a conversão seja usada pelo sistema operacional atual ou se a conversão for compatível com a atual máquina. |
| RSTOBJ FRCOBJCVN(\*SIM \*TODOS) | Todos os objetos são convertidos independentemente de seu formato atual e compatibilidade de máquina, incluindo objetos compatíveis no formato atual. |

A opção RSTOBJ FRCOBJCVN(\*SIM \*RQD) faz com que qualquer módulo incompatível e objetos de programa restaurados sejam imediatamente convertidos, em vez de subsequentemente retraduzidos durante a primeira ativação. Você deve considerar o uso dessa opção se não quiser alterar o valor do sistema QFRCCVNRST, mas quiser garantir que todos os objetos incompatíveis que sejam restaurados por este comando é convertido imediatamente.

**Criar opções**

Quando você cria objetos de módulo e programa, o comportamento padrão de usar todos os recursos disponíveis na máquina atual geralmente é desejável. No entanto, se você estiver criando software para distribuição em uma variedade de sistemas, convém ser mais conservador ao especificar quais recursos devem ser usados em seu programas. Por exemplo, se você vende um pacote de software que é executado na versão 6.1, talvez não queira que seus programas e programas de serviço sejam convertidos nas máquinas de seus clientes . Isso pode acontecer se o sistema em que você cria contiver mais recursos do que os de alguns de seus clientes. A conversão em máquinas de clientes pode ser indesejável devido aos seguintes motivos:

v Os clientes podem preferir não incorrer no tempo gasto convertendo os objetos (durante a restauração ou durante a primeira ativação)

v Os programas convertidos contêm instruções de hardware diferentes das dos programas originais. Embora a funcionalidade do programa permaneça a mesma, isso pode afetar seus processos de suporte ao cliente.

Para evitar a conversão, você pode especificar que seus programas sejam criados somente com recursos comuns a todos os sistemas suportados pela versão de destino. Você pode conseguir isso de uma das seguintes maneiras:

v Especificando a opção CodeGenTarget Licensed Internal Code (LICOPT) ao criar seus objetos de módulo e programa.

v Especificando o comando CodeGenTarget LICOPT em um programa de alteração (CHGPGM) ou Programa de serviço de alteração (CHGSRVPGM ). v Definindo uma variável de ambiente para controlar quais recursos são usados.

**O CodeGenTarget LICOPT**

Você pode especificar o parâmetro LICOPT (Opções de código interno licenciado) em muitos dos comandos de criação de módulo e programa, como Criar módulo C (CRTCMOD) e Criar programa C vinculado (CRTBNDC). O parâmetro LICOPT direciona o conversor otimizador para usar ou evitar certas opções quando cria as instruções de hardware para objetos. Você pode usar a opção LICOPT CodeGenTarget para fornecer controle sobre os recursos que são selecionados pela geração de código adaptável. Consulte "CodeGenTarget" na página 138 para todos os valores possíveis desta opção.

Ao selecionar a opção CodeGenTarget=Common ao criar seu módulo e objetos de programa, você pode garantir que seu produto de software não seja convertido quando for restaurado em máquinas do cliente , ou quando é usado pela primeira vez por seus clientes. No entanto, você pode estar desistindo de algumas melhorias de desempenho em potencial em máquinas com mais recursos de hardware.

A opção CodeGenTarget=Power6 pode ser útil se você tiver uma máquina mais antiga que você usa para compilações de software, mas deseja implantar seu software em máquinas que usam processadores POWER6. Essa opção faz com que o tradutor otimizador use os recursos do POWER6, mesmo que eles não estejam disponíveis na máquina atual. Os programas podem então ser implantados em seus sistemas POWER6 sem qualquer conversão durante o processamento de restauração ou a primeira ativação. No entanto, os programas não podem ser ativados na máquina de compilação sem serem convertidos. Você precisa garantir que os programas não sejam ativados na máquina de compilação; caso contrário, os recursos do POWER6 são removidos dos programas.

Depois de criar um objeto de programa, você pode perceber que você deve ter especificado quais recursos têm permissão para ser usado no objeto. Em vez de recriar o programa, você pode usar o comando Change Program (CHGPGM) ou Change Service Program (CHGSRVPGM) e especificar o parâmetro CodeGenTarget LICOPT nesse comando. O programa é convertido e o tradutor otimizador usa apenas os recursos que você especificou.

| Atualmente, os novos recursos estão disponíveis apenas para o nível POWER6 e POWER7 do PowerPC AS

| arquitetura. No IBM i 6.1, os recursos do POWER6 são selecionados com CodeGenTarget=Power6 e são

| não selecionado com CodeGenTarget=Legacy ou CodeGenTarget=Common. O efeito de

| CodeGenTarget=Current depende se você está executando no hardware POWER6 ou não. No IBM i

| 7.1, os recursos do POWER6 são selecionados com CodeGenTarget=Power6 e não são selecionados com

| CodeGenTarget=Legacy ou CodeGenTarget=Comum. As características do POWER7 são selecionadas com

| CodeGenTarget=Power7 e não são selecionados com CodeGenTarget=Power6, CodeGenTarget=Legacy ou

| CodeGenTarget=Comum. O efeito de CodeGenTarget=Current depende se você está executando no | POWER7, POWER6 ou hardware anterior. Observe que os recursos do POWER7 incluem todos os recursos do

| POTÊNCIA6.

Os recursos associados ao hardware POWER6 incluem: v Uma unidade decimal de ponto flutuante de hardware v Suporte de hardware eficiente para manipulação de ponteiro ILE

| Os recursos associados ao hardware POWER7 incluem:

| v Uma série de novas instruções que podem acelerar certos cálculos, como conversões entre | valores inteiros e de ponto flutuante.

Para um objeto de módulo, use o comando Display Module (DSPMOD) com DETAIL(\*BASIC) para ver as opções LICOPT aplicadas a esse módulo. Para um programa ou objeto de programa de serviço, as opções LICOPT são associadas a cada módulo do programa. Use o comando Programa de exibição (DSPPGM) ou Programa de serviço de exibição (DSPSRVPGM) com DETAIL(\*MODULE ) e especifique a opção 5 para os módulos que deseja exibir. O valor das opções de Código Interno Licenciado em uma dessas telas de exibição pode conter CodeGenTarget=model para alguns modelos. Isso indica que o LICOPT é usado para

substitua o comportamento padrão quando o módulo foi criado, especificando diretamente o LICOPT ou definindo a variável de ambiente QIBM\_BN\_CREATE\_WITH\_COMMON\_CODEGEN. Se tal LICOPT não estiver presente, o comportamento padrão não foi substituído.

Para obter mais informações sobre como exibir as opções de Código Interno Licenciado, consulte "Exibindo Opções de Código Interno Licenciado do Módulo e do Programa ILE " na página 142.

**Variável** **de ambiente** QIBM\_BN\_CREATE\_WITH\_COMMON\_CODEGEN

| Para compilações grandes, pode não ser conveniente especificar o CodeGenTarget LICOPT para todos os módulos e

| comandos de criação de programas. Além disso, o comando que você usa para criar objetos pode não oferecer suporte ao

| Parâmetro LICOPT. Nessas situações, você pode usar o

| QIBM\_BN\_CREATE\_WITH\_COMMON\_CODEGEN variável de ambiente para definir o código adaptável | comportamento de geração para suas compilações.

Para trabalhar com variáveis de ambiente, use o comando Trabalhar com variável de ambiente (WRKENVVAR).

Os valores possíveis para a variável de ambiente QIBM\_BN\_CREATE\_WITH\_COMMON\_CODEGEN estão na tabela a seguir.

|  |  |
| --- | --- |
| **Valor** | **Significado** |
| 0 | Ao criar novos módulos, use LICOPT CodeGenTarget conforme especificado. Se CodeGenTarget não for especificado, use CodeGenTarget=Current. (Este é o padrão.) |
| 1 | Ao criar novos módulos, use LICOPT CodeGenTarget conforme especificado. Se CodeGenTarget não for especificado, use CodeGenTarget=Common. |
| 2 | Ao criar novos módulos, sempre use CodeGenTarget=Common, substituindo qualquer CodeGenTarget LICOPT que possa ter sido fornecido. |

v Valor 0 denota o comportamento padrão.

v O valor 1 é útil quando você deseja usar o subconjunto comum de recursos na maioria das partes, mas gostaria de substituir isso para partes individuais. v Valor 2 é útil quando você precisa garantir que todas as peças sejam criadas com o subconjunto comum de recursos.

**Nota:** Esta variável de ambiente aplica-se apenas quando você cria novos módulos. Ele não se aplica a comandos de alteração, como o Programa de Alteração (CHGPGM). Também não se aplica durante a conversão de objetos.

**Exibindo informações do ACG**

Você pode usar os comandos Display Module (DSPMOD), Display Program (DSPPGM) e Display Service Program (DSPSRVPGM) para determinar se o módulo e os objetos de programa são compatíveis com a máquina atual e se o CodeGenTarget LICOPT é usado para substituir o comportamento ACG padrão.

#### Compatibilidade de objetos

Usar o Programa de Exibição (DSPPGM), o Programa de Serviço de Exibição (DSPSRVPGM) ou o Módulo de Exibição

(DSPMOD) com DETAIL(\*BASIC) para determinar se o módulo ou objeto de programa é capaz de ser executado corretamente sem primeiro ser convertido. O valor do campo Conversão necessária é \*SIM ou \*NÃO. Se o valor de Conversão necessário for \*SIM para um objeto de programa, o programa será convertido na primeira vez que for ativado. Como alternativa, você pode usar CHGPGM FRCCRT(\*YES) ou Start Object Conversion (STROBJCVN) para forçar a conversão a ocorrer em um momento mais conveniente. Consulte "Otimizando programas compatíveis" na página 148 para obter mais informações sobre quando esses dois comandos podem ser usados.

Se o valor do campo Conversão obrigatória for \*SIM para um objeto de módulo e o objeto de módulo não estiver em um formato mais antigo (consulte \*FORMAT na tabela a seguir), vinculando isso objeto de módulo em um programa ou programa de serviço resulta em um objeto de programa que também requer conversão. Esse resultado é útil quando você está criando programas para implantação em um sistema que tem mais recursos do que o sistema de compilação. Se o valor do campo Conversão necessária for \*NO, o objeto de programa ou módulo estará pronto para uso imediato.

O motivo pelo qual um módulo ou objeto de programa requer conversão pode ser determinado exibindo o campo Detalhes da conversão. Este campo tem um dos seguintes valores.

|  |  |
| --- | --- |
| **Valor** | **Significado** |
| \*FORMATO | O objeto não é compatível com a máquina atual. O objeto está em um formato mais antigo. (Para por exemplo, objetos criados para versões anteriores à 6.1 estão em um formato diferente dos objetos criados para a versão 6.1.) A associação de um objeto de módulo em um formato mais antigo faz com que o módulo acoplado seja convertido. |
| \*CARACTERÍSTICA | O objeto não é compatível com a máquina atual. O formato do objeto é compatível com a máquina atual, mas o objeto usa pelo menos um recurso que não é suportado pela máquina atual. |
| \*COMPAT | O objeto é compatível com a máquina atual. O formato do objeto é compatível com a máquina atual e todos os recursos que o objeto usa são implementados pela máquina atual. No entanto, o objeto usa pelo menos um recurso que não é suportado pelo nível comum de hardware suportado pela versão, lançamento e nível de modificação que o objeto foi criado para. |
| \*COMUM | O objeto é compatível com a máquina atual. O formato do objeto é compatível e todos os recursos que o objeto usa são implementados pelo nível comum de hardware suportado pelo nível de versão, lançamento e modificação que o objeto foi criado para. |

**Considerações** **de** **liberação para versão**

O comportamento da geração de código adaptável é determinado pela versão, liberação e nível de modificação de destino escolhidos ao criar um módulo ou objeto de programa. O ACG só é suportado em sistemas com a versão 6.1 ou posterior. Se o CodeGenTarget LICOPT for especificado para um objeto com uma versão de destino anterior, o LICOPT será tolerado, mas não terá efeito sobre o código gerado para o objeto.

A versão de destino, a versão e o nível de modificação também regem o significado de

LICOPT('CodeGenTarget=Comum'). Quando este LICOPT é selecionado, isso significa que o tradutor otimizador deve usar o conjunto de recursos disponíveis em todas as máquinas suportadas pela versão de destino, que pode diferir do na qual o objeto de programa ou módulo foi criado. Portanto, você pode usar uma máquina de compilação executando uma versão mais recente do sistema operacional para criar código comum que é executado em cada computador que executa uma versão anterior do sistema operacional, desde que esta versão anterior não seja anterior à 6.1.

**Nota:** Você não pode usar o tradutor de otimização que está em uma versão anterior para aproveitar todos os recursos comuns a máquinas em execução em uma versão mais recente.

**Otimizando programas compatíveis**

Quando objetos compatíveis são restaurados em uma máquina, eles normalmente não são convertidos, a menos que você tenha especificado o parâmetro FRCOBJCVN(\*YES \*ALL) no comando Restore. Isso significa que os objetos podem não aproveitar ao máximo os recursos da máquina. Alguns recursos podem não ter sido usados pelo tradutor otimizador. Talvez você queira atualizar seus objetos de módulo e programa para aproveitar ao máximo sua máquina.

Você precisa saber se o módulo e os objetos de programa têm dados completos de criação disponíveis. Use o comando Display Module (DSPMOD), Display Program (DSPPGM) ou Display Service Program (DSPSRVPGM) com DETAIL(\*BASIC) para ver isso.

v Se todos os dados de criação estiverem disponíveis e observáveis, o campo Todos os dados de criação será \*SIM. Você pode usar os comandos Módulo de Alteração (CHGMOD), Programa de Alteração (CHGPGM) e Programa de Serviço de Alteração (CHGSRVPGM) com FRCCRT(\*YES) para forçar a recriação dos objetos.

v Se todos os dados de criação estiverem disponíveis, mas nem todos forem observáveis, o campo Todos os dados de criação será \*UNOBS. Você pode forçar uma conversão desses objetos durante uma operação de restauração especificando o parâmetro FRCOBJCNV(\*YES \*ALL) no comando Restore.

| v Se nem todos os dados de criação estiverem disponíveis, o campo Todos os dados de criação será \*NO. Você não pode fazer nada para mudar o | objetos. Se os programas não tiverem dados de criação dessa maneira, eles deverão ser recriados da origem para serem executados em | 6.1 e versões posteriores.

**ACG e partições** **lógicas**

Se você tiver um sistema com várias partições lógicas, poderá configurar uma partição para imitar um sistema com um processador diferente daquele que você instalou fisicamente . Para fins de geração de código adaptável, tal partição é tratada como se estivesse sendo executada no processador imitado. Quaisquer recursos em um programa que não estejam disponíveis no processador imitado forçam o programa a exigir a conversão para ser executado nesse processador, mesmo que esses recursos estejam disponíveis no processador físico subjacente.

**150 Conceitos** ILE IBM i 7.1

**Capítulo 14.**  **Sincronização de armazenamento**  compartilhado

O armazenamento compartilhado fornece um meio eficiente para a comunicação entre dois ou mais threads em execução simultânea. Este capítulo discute uma série de problemas relacionados ao armazenamento compartilhado. O foco principal está nos problemas de sincronização de dados que podem surgir ao acessar o armazenamento compartilhado e como superá-los .

Embora não sejam exclusivos do ILE, os problemas de programação associados ao armazenamento compartilhado são mais prováveis de ocorrer em linguagens ILE do que em linguagens MI originais. Isso se deve ao suporte mais amplo para interfaces de programação de aplicativos de multiprogramação no ILE.

**Armazenamento** compartilhado

O termo *armazenamento compartilhado*, no que se refere a esta discussão, refere-se a quaisquer dados de espaço que são acessados a partir de dois ou mais threads. Essa definição inclui qualquer armazenamento diretamente acessível até seus bytes individuais e pode incluir as seguintes classes de armazenamento: v objetos de espaço MI v Espaços primários associados de outros MI objetos v segmentos de memória compartilhada POSIX

v Espaços de processo implícitos : armazenamento automático , armazenamento estático e armazenamento de pilha baseado em ativação v Teraspace

O sistema considera esses espaços, independentemente da longevidade de sua existência, como armazenamento compartilhado quando acessados por múltiplos threads capazes de processamento simultâneo.

**Armadilhas de armazenamento** compartilhado

Ao criar aplicativos que aproveitam o armazenamento compartilhado, você precisa evitar dois tipos de problemas que podem resultar em valores de dados imprevisíveis:  *condições* de *corrida* e armazenamento *problemas de pedidos de acesso*.

v Uma condição de corrida existe quando diferentes resultados do programa são possíveis devido exclusivamente ao tempo relativo de dois ou mais threads cooperantes.

Você pode evitar condições de corrida sincronizando o processamento dos threads concorrentes para que eles interajam de maneira previsível e bem comportada. Embora o foco deste documento esteja na sincronização de armazenamento, as técnicas de sincronização da execução de threads e de sincronização de armazenamento se sobrepõem em grande medida. Por causa disso, os problemas de exemplo discutidos mais adiante neste capítulo abordam brevemente as condições de corrida.

v Problemas de ordenação de acesso ao armazenamento também são conhecidos como sincronização de armazenamento ou problemas de consistência de memória. Esses problemas resultam quando dois ou mais threads cooperantes dependem de um pedido específico de atualizações para o armazenamento compartilhado e seus respectivos acessos aos acessos ao armazenamento são não sincronizado. Por exemplo, um thread pode armazenar valores em duas variáveis compartilhadas e outro thread tem uma dependência implícita de observar essas atualizações em uma determinada ordem.

Você pode evitar problemas de ordenação de acesso ao armazenamento compartilhado garantindo que o sistema execute ações de sincronização de armazenamento para os threads que lêem e gravam no armazenamento compartilhado . Algumas dessas ações são descritas nos tópicos a seguir.

© Direitos autorais IBM Corp. 1997, 2010

**Pedido de acesso ao**  armazenamento compartilhado

Quando os threads compartilham armazenamento, não existe garantia de que os acessos ao armazenamento compartilhado (leituras e gravações) executados por um thread serão observados nessa ordem específica por outros threads. Você pode evitar isso fazendo com que alguma forma de sincronização de armazenamento explícita seja executada pelos threads que estão lendo ou gravando no armazenamento compartilhado.

A sincronização de armazenamento é necessária quando dois ou mais threads tentam acesso simultâneo ao armazenamento compartilhado, e a semântica da lógica dos threads requer alguma ordenação nos acessos de armazenamento compartilhado . Quando a ordem em que as atualizações de armazenamento compartilhado são observadas não é importante, nenhuma sincronização de armazenamento é necessária. Um determinado thread sempre observará suas próprias atualizações de armazenamento (para armazenamento compartilhado ou não compartilhado) em ordem. Todos os threads que acessam locais de armazenamento compartilhado *sobrepostos* observarão esses acessos na mesma ordem.

Considere o exemplo simples a seguir, que ilustra como as condições de corrida e os problemas de pedidos de acesso ao armazenamento podem levar a resultados imprevisíveis.

int volátil X = 0; int volátil Y = 0;

Thread A Thread B

-------------- -------------

Y = 1; impressão (X);

X = 1; impressão(Y);

A tabela abaixo resume os possíveis resultados que são impressos por B.

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Tipo de problema** | **Explicação** |
| 0 | 0 | Condição da Corrida | O Thread B lê as variáveis antes das modificações do Thread A. |
| 0 | 1 | Condição da Corrida | O Thread B observou a atualização para Y, mas imprimiu X antes de observar a atualização do Thread A. |
| 1 | 1 | Condição da Corrida | O Thread B lê ambas as variáveis após as atualizações do Thread A. |
| 1 | 0 | Pedido de acesso ao armazenamento | O Thread B observou a atualização para X, mas ainda não tinha visto a atualização do Thread A para Y. Sem ações explícitas de sincronização de dados, esse tipo de acesso ao armazenamento fora de sequência pode ocorrer. |

**Exemplo de Problema 1: Um Escritor, Muitos Leitores**

Normalmente, o potencial de acessos de armazenamento compartilhado fora de ordem não afeta a correção da lógica de programa multithread. No entanto, em alguns casos, a ordem em que os threads veem as atualizações de armazenamento de outros threads é vital para a correção do programa.

Considere um cenário típico que requer alguma forma de sincronização de dados explícita. É quando o estado de um local de armazenamento compartilhado é usado (por convenção na lógica de um programa) para controlar o acesso a um segundo armazenamento compartilhado (não sobreposto) localização. Por exemplo, suponha que um thread inicialize alguns dados compartilhados (DATA). Além disso, suponha que o thread defina um sinalizador compartilhado (FLAG) para indicar a todos os outros threads que os dados compartilhados estão inicializados.

|  |  |
| --- | --- |
| Inicializando thread | Todos os outros tópicos |
| ------------------- | --------------------------- |
| DADOS = 10 | até que FLAG tenha valor 1 |
| BANDEIRA = 1 | usar DADOS |

Nesse caso, os threads de compartilhamento devem impor uma ordem nos acessos de armazenamento compartilhado. Caso contrário, outros threads podem exibir as atualizações de armazenamento compartilhado do thread inicializador fora de ordem. Isso pode permitir que alguns ou todos os outros threads leiam um valor não inicializado de DATA.

**152 Conceitos** de ILE IBM i 7.1

#### Exemplo 1 Solução

Um método preferencial para resolver o problema no exemplo acima é evitar a dependência entre os dados e os valores de sinalizador. Você pode fazer isso usando um esquema de sincronização de thread mais robusto. Embora você possa empregar muitas das técnicas de sincronização de thread, uma que se presta bem a esse problema é um semáforo.

Para que a seguinte lógica seja apropriada, você deve assumir o seguinte: v O programa criou o semáforo antes de iniciar os threads cooperantes. v O programa inicializou o semáforo a uma contagem de 1.

|  |  |
| --- | --- |
| Inicializando thread | Todos os outros tópicos |
| ------------------- | --------------------------- |
| DADOS = 10 | Aguarde até que a contagem de semáforos atinja 0 |
| Semáforo de decremento | usar DADOS |

**Ações de sincronização de**  armazenamento

Quando uma ordenação de acessos de armazenamento compartilhado é necessária, todos os threads que exigem a imposição de um pedido devem executar uma ação explícita para sincronizar os acessos de armazenamento compartilhado. Essas ações são chamadas de ações de  *sincronização de armazenamento*.

Uma ação de sincronização executada por um thread garante que os acessos ao armazenamento compartilhado que aparecem antes da ação de sincronização no fluxo lógico do thread sejam concluídos antes dos acessos que aparecem no fluxo lógico do código após a ação de sincronização. Isso é do ponto de vista de outros threads em suas ações de sincronização. Em outras palavras, se um thread executa duas gravações em dois locais compartilhados e uma ação de sincronização separa essas gravações, o sistema realiza o seguinte: A primeira gravação é garantido para estar disponível para outros threads em ou antes de suas próximas ações de sincronização, e o mais tardar no ponto em que a segunda gravação se torna disponível.

Quando duas leituras de dois locais compartilhados são separadas por uma ação de sincronização de armazenamento, a segunda leitura lê um valor não menos atual do que a primeira leitura. Isso só é verdadeiro quando outros threads impõem uma ordem ao gravar no armazenamento compartilhado.

As seguintes ações de sincronização de thread também são ações de sincronização de armazenamento:

|  |  |  |
| --- | --- | --- |
| **Mecanismo** | **Ação** de sincronização | **Primeiro disponível em VRM** |
| Bloqueios de objeto | Bloquear, Desbloquear | Todo |
| Bloqueios de localização de espaço | Bloquear, Desbloquear | Todo |
| Mutex | Bloquear, Desbloquear | V3R1M0 |
| Semáforos | Postar, Esperar | V3R2M0 |
| Condições Pthread | Esperar, Sinal, Transmitir | V4R2M0 |
| Filas de dados | Enfileirar, Remover fila | Todo |
| Filas de mensagens | Enfileirar, Remover fila | V3R2M0 |
| Comparar e trocar | Loja bem-sucedida para segmentar | V3R1M0 |
| Verificar valor de bloqueio (CHKLKVAL) | Loja bem-sucedida para segmentar | V5R3M0 |
| Valor de bloqueio claro (CLRLKVAL) | Sempre | V5R3M0 |

Além disso, a seguinte instrução MI constitui uma ação de sincronização de armazenamento, mas não é utilizável para sincronizar threads:

|  |  |  |
| --- | --- | --- |
| **Mecanismo** | **Ação** de sincronização | **Primeiro disponível em VRM** |
| Instrução SYNCSTG MI | Sempre | V4R5M0 |

Capítulo 14. Sincronização de armazenamento compartilhado

Lembre-se: para impor completamente a ordenação de acesso ao armazenamento compartilhado entre dois ou mais threads, todos os threads que dependem da ordem de acesso devem usar as ações de sincronização apropriadas. Isso é verdade tanto para os leitores quanto para os escritores dos dados compartilhados. Este acordo entre leitores e gravadores garante que a ordem dos acessos permanecerá inalterada por quaisquer otimizações empregadas pela máquina subjacente.

**Exemplo de Problema 2: Dois** Escritores  **ou Leitores Contendores**

Outro problema comum que requer sincronização adicional é aquele em que dois ou mais threads tentam impor um protocolo de bloqueio informal, como no exemplo abaixo. Neste exemplo, dois threads manipulam dados no armazenamento compartilhado. Ambos os threads tentam repetidamente ler e gravar dois itens de dados compartilhados, usando um sinalizador compartilhado na tentativa de serializar acessos.

|  |  |
| --- | --- |
| Segmento A | Segmento B |
| -------------------------------------- | -------------------------------------- |
| /\* Faça algum trabalho nos dados compartilhados \*/ | /\* Faça algum trabalho nos dados compartilhados \*/ |
| for (int i=0; i<10; ++i) { | for (int i=0; i<10; ++i) { |
| /\* Aguarde até que o sinalizador bloqueado esteja limpo \*/ | /\* Aguarde até que o sinalizador bloqueado esteja limpo \*/ |
| while (bloqueado == 1) { | while (bloqueado == 1) { |
| sono(1); | sono(1); |
| } | } |
| bloqueado = 1; /\* Defina o bloqueio \*/ | bloqueado = 1; /\* Definir o bloqueio \*/ |
| /\* Atualizar os dados compartilhados \*/ | /\* Atualizar os dados compartilhados \*/ |
| dados1 += 5; | dados1 += 4; |
| dados2 += 10; | dados2 += 6; |
| bloqueado = 0; /\* Limpe o cadeado \*/ | bloqueado = 0; /\* Limpe o cadeado \*/ |
| } | } |

Este exemplo ilustra ambas as nossas armadilhas de memória compartilhada.

#### Condições da Corrida

O protocolo de bloqueio usado aqui não contornou as condições de corrida de dados. Ambos os trabalhos poderiam ver simultaneamente que o sinalizador bloqueado é claro e, portanto, ambos se enquadram na lógica que atualiza os dados. Nesse ponto, não há garantia de quais valores de dados serão lidos, incrementados e gravados – permitindo muitos resultados possíveis.

#### Preocupações com pedidos de acesso ao armazenamento

Ignore, por um momento, a condição de corrida mencionada acima. Observe que a lógica usada por ambos os trabalhos para atualizar o bloqueio e os dados compartilhados contém suposições sobre a ordem implícita das atualizações de campo. Especificamente, há uma suposição na parte de cada thread de que o outro thread observará que o sinalizador bloqueado foi definido como 1 antes de observar alterações nos dados. Além disso, presume-se que cada thread observará a alteração dos dados antes de observar o valor do sinalizador bloqueado de zero. Como observado anteriormente nesta discussão, essas suposições não são válidas.

#### Exemplo 2 Solução

Para evitar a condição de corrida e impor a ordem de armazenamento , você deve serializar os acessos aos dados compartilhados por um dos mecanismos de sincronização enumerados acima. Este exemplo, em que vários threads estão competindo por um recurso compartilhado, se presta bem a alguma forma de bloqueio. Uma solução empregando um bloqueio de localização de espaço será discutida, seguida por uma solução alternativa usando o Valor de Bloqueio de Verificação e o Valor de Bloqueio de Limpeza.

|  |  |
| --- | --- |
| THREAD A | THREAD B |
| -------------------------------------- | ------------------------------------ |
| for (i=0; i<10; ++i) { | for (i=0; i<10; ++i) { |
| /\* Obtenha um bloqueio exclusivo no compartilhamento | /\* Obtenha um bloqueio exclusivo no compartilhamento |
| dados. Entramos em um estado de espera até | dados. Entramos em um estado de espera até |
| o bloqueio é concedido. \*/ | o bloqueio é concedido. \*/ |
| locksl( LOCK\_LOC, \_LENR\_LOCK); | locksl( LOCK\_LOC, \_LENR\_LOCK); |
| /\* Atualizar os dados compartilhados \*/ | /\* Atualizar os dados compartilhados \*/ |
| dados1 += 5; | dados1 += 4; |
| dados2 += 10; | dados2 += 6; |

**154 Conceitos** de ILE IBM i 7.1

/\* Desbloqueie os dados compartilhados \*//\* Desbloqueie os dados compartilhados \*/

unlocksl( LOCK\_LOC, \_LENR\_LOCK); unlocksl( LOCK\_LOC, \_LENR\_LOCK);

} }

Restringir o acesso aos dados compartilhados com um bloqueio garante que apenas um thread poderá acessar os dados por vez. Isso resolve a condição da corrida. Essa solução também resolve os problemas de pedido de acesso ao armazenamento, uma vez que não há mais uma dependência de pedidos entre dois locais de armazenamento compartilhados.

#### Solução alternativa: Usando o valor de bloqueio de verificação / Valor de bloqueio de limpeza

Os bloqueios de localização de espaço, como os usados na primeira solução, estão cheios de recursos que não são necessários neste exemplo simples. Por exemplo, os bloqueios de localização de espaço suportam um valor de tempo limite que permitiria que o processamento fosse retomado se não fosse possível adquirir o bloqueio dentro de algum período de tempo. Os bloqueios de locais de espaço também oferecem suporte a várias combinações de bloqueios compartilhados. Esses são recursos importantes, mas vêm ao preço de alguma sobrecarga de desempenho.

Uma alternativa é usar o Valor de Bloqueio de Verificação e  *o Valor de Bloqueio de Limpeza*. Juntas, essas duas instruções MI fornecem uma maneira de implementar um protocolo de bloqueio muito simples e rápido, especialmente se não houver muita contenção no bloqueio.

Nesta solução, o sistema usa CHKLKVAL para tentar adquirir a fechadura. Se essa tentativa falhar (porque o sistema encontrou o bloqueio já em uso), o thread aguardará por um tempo e tentará novamente, repetindo até que o bloqueio esteja adquirido. Depois de atualizar os dados compartilhados, o sistema usará o CLRLKVAL para liberar o bloqueio. Neste exemplo, suponha que, além dos itens de dados compartilhados, os threads também compartilham o endereço de um local de 8 bytes. Esse código refere-se a esse local como variável LOCK. Além disso, suponha que o bloqueio foi inicializado para zero, seja por meio de inicialização estática ou alguma inicialização sincronizada anterior.

|  |  |
| --- | --- |
| THREAD A | THREAD B |
| -------------------------------- | --------------------------------- |
| /\* Faça algum trabalho nos dados compartilhados \*/ | /\* Faça algum trabalho nos dados compartilhados \*/ |
| for (i=0; i<10; ++i) { | for (i=0; i<10; ++i) { |
| /\* Tente adquirir o bloqueio usando | /\* Tente adquirir o bloqueio usando |
| CHKLKVAL. Por convenção, valor de uso | CHKLKVAL. Por convenção, valor de uso |
| 1 para indicar bloqueado, 0 para indicar | 1 para indicar bloqueado, 0 para indicar |
| desbloqueado. \*/ | desbloqueado. \*/ |
| while ( \_CHKLKVAL(&LOCK, 0, 1) == 1) { | while ( \_CHKLKVAL(&LOCK, 0, 1) == 1) { |
| sleep(1); /\* aguarde um pouco e tente novamente \*/ | sleep(1); /\* aguarde um pouco e tente novamente \*/ |
| } | } |
| /\* Atualizar os dados compartilhados \*/ | /\* Atualizar os dados compartilhados \*/ |
| dados1 += 5; | dados1 += 4; |
| dados2 += 10; | dados2 += 6; |
| /\* Desbloqueie os dados compartilhados. Utilização de | /\* Desbloqueie os dados compartilhados. Utilização de |
| CLRLKVAL garante outros trabalhos/threads | CLRLKVAL garante outros trabalhos/threads |
| consulte a atualização dos dados compartilhados antes de | consulte a atualização dos dados compartilhados antes de |
| liberação da fechadura. \*/ | liberação da fechadura. \*/ |
| \_CLRLKVAL(&LOCK, 0); | \_CLRLKVAL(&LOCK, 0); |
| } | } |

Aqui, os threads usam o Valor de Bloqueio de Verificação para executar um teste e uma atualização sem corrida da variável de bloqueio e o Valor de Bloqueio de Limpeza para redefinir a variável de bloqueio para o estado desbloqueado. Isso resolve a condição de corrida experimentada nos fragmentos do problema original. Ele também aborda o problema de pedido de acesso ao armazenamento. Como observado anteriormente, usados dessa maneira, o Check Lock Value e o Clear Lock Value são ações de sincronização. O uso do Valor de Bloqueio de Verificação para definir o bloqueio antes de ler os dados compartilhados garante que os threads leiam os dados atualizados mais recentemente. O uso de Limpar Valor de Bloqueio para limpar o bloqueio após a atualização dos dados compartilhados garante que as atualizações estejam disponíveis para leituras subsequentes por qualquer thread após a sua próxima ação de sincronização.

Capítulo 14. Sincronização de armazenamento compartilhado

**156 Conceitos** de ILE IBM i 7.1

**Apêndice A. Listagem de saída do** comando  **CRTPGM, CRTSRVPGM, UPDPGM ou UPDSRVPGM**

Este apêndice mostra exemplos de listagens de fichários e explica erros que podem ocorrer como resultado do uso da linguagem do fichário.

**Listagem do fichário**

As listagens de fichários para os comandos Criar Programa (CRTPGM), Criar Programa de Serviço (CRTSRVPGM), Programa de Atualização (UPDPGM) e Programa de Serviço de Atualização (UPDSRVPGM) são quase idênticas. Este tópico apresenta uma listagem de fichário do comando CRTSRVPGM usado para criar o programa de serviço FINANCEIRO em "Binder Language Examples" na página 71.

Três tipos de listagens podem ser especificados no parâmetro detail (DETAIL) dos comandos CRTPGM, CRTSRVPGM, UPDPGM ou UPDSRVPGM :

\*BÁSICO

\*PROLONGADO

\*CHEIO

**Listagem Básica**

Se você especificar DETAIL(\*BASIC) no comando CRTPGM, CRTSRVPGM, UPDPGM ou UPDSRVPGM, a listagem consistirá no seguinte:

v Os valores especificados no comando CRTPGM, CRTSRVPGM, UPDPGM ou UPDSRVPGM v Uma breve tabela de resumo

v Os dados que mostram o tempo que algumas partes do processo de vinculação levaram para serem concluídas A Figura 46, a Figura 47 e a Figura 48 na página 159 mostram essas informações.

© Direitos autorais IBM Corp. 1997, 2010

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Criar programa de serviço | | |  |  | 1 |
| Programa de serviço . . . . . . . . . . . . . . . . . . . . . . : | FINANCEIRO |  |
| Biblioteca............... . . . . . : | MYLIB |  |  |  |  |
| Exportação . . . . . . . . . . . . . . . . . . . . . . . . . . : | \*SRCFILE |  |  |  |  |
| Exportar o ficheiro de origem . . . . . . . . . . . . . . . . . : | QSRVSRC |  |  |  |  |
| Biblioteca............... . . . . . : | MYLIB |  |  |  |  |
| Membro da fonte de exportação . . . . . . . . . . . . . . . . . . . : | \*SRVPGM |  |  |  |  |
| Grupo de activação . . . . . . . . . . . . . . . . . . : | \*CHAMADOR |  |  |  |  |
| Permitir atualização . . . . . . . . . . . . . . . . . . . . . . . . . . . . : | \*SIM |  |  |  |  |
| Permitir a atualização do nome da biblioteca \*SRVPGM vinculada . . . . . : | \*NÃO |  |  |  |  |
| Opções de criação . . . . . . . . . . . . . . . . . . . . : | \*GEN | \*NODUPPROC | \*NODUPVAR | \*DUPWARN |  |
| Detalhes da listagem . . . . . . . . . . . . . . . . . . . . . . . . : | \*CHEIO |  |  |  |  |
| Perfil do utilizador . . . . . . . . . . . . . . . . . . . . . . . . : | \*UTILIZADOR |  |  |  |  |
| Substituir o programa de serviço existente . . . . . . . : | \*SIM |  |  |  |  |
| Libertação alvo . . . . . . . . . . . . . . . . . . . . . . . . . . : | \*ATUAL |  |  |  |  |
| Permitir a reinicialização . . . . . . . . . . . . . . . . : | \*NÃO |  |  |  |  |
| Modelo de armazenamento . . . . . . . . . . . . . . . . . . . . . : | \*SNGLVL |  |  |  |  |
| Otimização de argumentos . . . . . . . . . . . . : | \*NÃO |  |  |  |  |
| Análise interprocessual . . . . . . . . . . . : | \*NÃO |  |  |  |  |
| Ficheiro de controlo IPA . . . . . . . . . . . . . . . . : | \*NENHUM |  |  |  |  |
| Autoridade. . . . . . . . . . . . . . . . . . . . . . : Texto . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  Biblioteca de módulos Biblioteca de módulos  DINHEIRO MYLIB CALCS MYLIB  PREÇOS MYLIB ACCTS MYLIB  Serviço  Ativação da biblioteca de programas  \*NENHUM  Ligação  Ativação da biblioteca de diretórios  \*NENHUM | \*LIBCRTAUT |  |  |  |  |
| *Figura 46. Valores especificados no comando CRTSRVPGM* | |
| Criar programa de serviço  Breve Tabela de Resumo  Procedimentos de entrada no programa . . . . . . . . . . : 0  Múltiplas definições fortes . . . . . : 0  Referências não resolvidas . . . . . . . . . . : 0 | | |  | Página | 3 |
| \* \* \* \* \*E N D O F B R I E F S U M M A R Y T A B L E | | | | \* \* \* \* \* |  |

*Figura 47. Breve Tabela de Resumo*

|  |  |  |  |
| --- | --- | --- | --- |
| Criar estatísticas de associação de programa de serviço | | Página | 23 |
| Tempo de CPU da coleção de símbolos . . . . . . . . . . . . . . . . . . . . . : | .018 |
| Tempo de CPU de resolução do símbolo . . . . . . . . . . . . . . . . . . . . . . . : | .006 |  |  |
| Tempo de CPU de resolução de diretório de vinculação . . . . . . . . . . . . . . : | .403 |  |  |
| Tempo de CPU de compilação da linguagem do fichário . . . . . . . . . . . . . . . : | .040 |  |  |
| Listando o tempo de CPU de criação . . . . . . . . . . . . . . . . . . . . . . . . . . . : | 1.622 |  |  |
| Tempo de CPU de criação de programa/programa de serviço . . . . . . . . : | .178 |  |  |
| Tempo total da CPU . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . : | 2.761 |  |  |
| Tempo total decorrido . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . : | 11.522 |  |  |
| \* \* \* \* \*E N D O F B I N D I N G  \*CPC5D0B - Programa de serviço FINANCEIRO criado na biblioteca MYLIB. | S T A T I S T I C S | \* \* \* \* \* |  |
| \* \* \* \* \*E N D O F C R E A T E S E R V I C E | P R O G R A M L I S T I N G\* \* \* \* \* \* \* | | |

*Figura 48. Estatísticas vinculativas*

**Listagem estendida**

Se você especificar DETAIL(\*EXTENDED) no comando CRTPGM, CRTSRVPGM, UPDPGM ou UPDSRVPGM , a listagem incluirá todas as informações fornecidas por DETAIL(\*BASIC) mais uma tabela de resumo estendida. O quadro de síntese alargado mostra o número de importações (referências) que foram resolvidas e o número de exportações (definições) transformadas . Para o comando CRTSRVPGM ou UPDSRVPGM, a listagem também mostra o idioma do fichário usado, as assinaturas geradas e quais importações

(referências) correspondeu a quais exportações (definições). A Figura 49, a Figura 50 na página 160 e a Figura 51 na página 161 mostram exemplos dos dados adicionais.

Criar Programa de Serviço Página 2

Tabela de resumo estendida

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Definições válidas . . . . . . . . : |  | 418 |  |  |  |
| Forte............ . . . : |  | 418 |  |  |  |
| Fraco . . . . . . . . . . . . . . . . . . . : |  | 0 |  |  |  |
| Referências resolvidas . . . . . . . : |  | 21 |  |  |  |
| Para definições fortes ... . . . . : |  | 21 |  |  |  |
| Para definições fracas . . . . . . . . : |  | 0 |  |  |  |
| \* \* \* \* \* E N D | O F | E X T E N D E D | S U M M A R Y | T A B L E | \* \* \* \* \* |

*Figura 49. Listagem resumida*  estendida

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criar programa de serviço  Listagem de informações do fichário | |  |  | 4 |
| Módulo...... . . . . . : Biblioteca MONEY . . . : MYLIB  Encadernado . . . . . . : \*SIM |  |
| Identificador Ref do Símbolo do Número | Tipo | Âmbito | Exportação | Chave |
| 00000001 Def principal | Proc | Módulo | Forte |  |
| 00000002 Def Montante | Proc | SrvPgm | Forte |  |
| 00000003 Def Pagamento | Proc | SrvPgm | Forte |  |
| 00000004 Ref 0000017F Q LE AG\_prod\_rc | Dados |  |  |  |
| 00000005 Ref 0000017E Q LE AG\_user\_rc | Dados |  |  |  |
| 00000006 Ref 000000AC \_C\_main | Proc |  |  |  |
| 00000007 Ref 00000180 Q LE leDefaultEh | Proc |  |  |  |
| 00000008 Ref 00000181 Q LE mhConversionEh | Proc |  |  |  |
| 00000009 Ref 00000125 \_C\_exception\_router  Módulo...... . . . . . : TAXAS Biblioteca . . . : MYLIB  Encadernado . . . . . . : \*SIM | Proc |  |  |  |
| Identificador Ref do Símbolo do Número | Tipo | Âmbito | Exportação | Chave |
| 0000000A Termo Def | Proc | SrvPgm | Forte |  |
| 0000000B Taxa de Def | Proc | SrvPgm | Forte |  |
| 0000000C Ref 0000017F Q LE AG\_prod\_rc | Dados |  |  |  |
| 0000000D Ref 0000017E Q LE AG\_user\_rc | Dados |  |  |  |
| 0000000E Ref 00000180 Q LE leDefaultEh | Proc |  |  |  |
| 0000000F Ref 00000181 Q LE mhConversionEh | Proc |  |  |  |
| 00000010 Ref 00000125 \_C\_exception\_router  Módulo...... . . . . . : Biblioteca CALCS . . . : MYLIB  Encadernado . . . . . . : \*SIM | Proc |  |  |  |
| Identificador Ref do Símbolo do Número | Tipo | Âmbito | Exportação | Chave |
| 00000011 Def Calc1 | Proc | Módulo | Forte |  |
| 00000012 Def Calc2 | Proc | Módulo | Forte |  |
| 00000013 Ref 0000017F Q LE AG\_prod\_rc | Dados |  |  |  |
| 00000014 Ref 0000017E Q LE AG\_user\_rc | Dados |  |  |  |
| 00000015 Ref 00000180 Q LE leDefaultEh | Proc |  |  |  |
| 00000016 Ref 00000181 Q LE mhConversionEh | Proc |  |  |  |
| 00000017 Ref 00000125 \_C\_exception\_router  Módulo...... . . . . . : Biblioteca ACCTS . . : MYLIB  Encadernado . . . . . . : \*SIM | Proc |  |  |  |
| Identificador Ref do Símbolo do Número | Tipo | Âmbito | Exportação | Chave |
| 00000018 Def OpenAccount | Proc | SrvPgm | Forte |  |
| 00000019 Def CloseAccount | Proc | SrvPgm | Forte |  |
| 0000001A Ref 0000017F Q LE AG\_prod\_rc | Dados |  |  |  |
| 0000001B Ref 0000017E Q LE AG\_user\_rc | Dados |  |  |  |
| 0000001C Ref 00000180 Q LE leDefaultEh | Proc |  |  |  |
| 0000001D Ref 00000181 Q LE mhConversionEh | Proc |  |  |  |
| 0000001E Ref 00000125 \_C\_exception\_router | Proc |  |  |  |

*Figura 50. Listagem de informações do fichário* (Parte 1 de 2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Programa de serviço . . . . . . : Biblioteca QC2SYS . . . : \*LIBL  Encadernado . . . . . . : \*NÃO | |  |  |  |  |
| Identificador Ref do Símbolo do Número | | Tipo | Âmbito | Exportação | Chave |
| 0000001F Sistema de def  Programa de serviço . . . . . . : Biblioteca QLEAWI . . . : \*LIBL Bound . . . . . . . | | Proc |  | Forte |  |
| Identificador Ref do Símbolo do Número | | Tipo | Âmbito | Exportação | Chave |
| 0000017E Def Q LE AG\_user\_rc | | Dados |  | Forte |  |
| 0000017F Def Q LE AG\_prod\_rc | | Dados |  | Forte |  |
| 00000180 Def Q LE leDefaultEh | | Proc |  | Forte |  |
| 00000181 Def Q LE mhConversionEh  *Figura 50. Listagem de informações do fichário (Parte 2 de 2)* | | Proc |  | Forte |  |
| Criar programa de serviço | |  |  | Página | 14 |
| Listagem de idioma do fichário | | |
| STRPGMEXP PGMLVL(\*ATUAL)  SÍMBOLO DE EXPORTAÇÃO ('Termo')  SÍMBOLO DE EXPORTAÇÃO ('Taxa')  SÍMBOLO DE EXPORTAÇÃO ('Quantidade')  SÍMBOLO DE EXPORTAÇÃO ('Pagamento')  SÍMBOLO DE EXPORTAÇÃO ('OpenAccount')  SÍMBOLO DE EXPORTAÇÃO ('CloseAccount')  ENDPGMEXP  Assinatura de exportação: 00000000ADCEFEE088738A98DBA6E723.  STRPGMEXP PGMLVL(\*PRV) SÍMBOLO DE EXPORTAÇÃO('termo')  SÍMBOLO DE EXPORTAÇÃO ('Taxa')  SÍMBOLO DE EXPORTAÇÃO ('Quantidade')  SÍMBOLO DE EXPORTAÇÃO ('Pagamento')  ENDPGMEXP  Assinatura de exportação: 0000000000000000000ADC89D09E0C6E7. |  | |
| \* \* \* \* \* E N D O F B I N D E R | L A N G U A G E | | L I S T I N G\* \* \* \* \* \* | |  |

*Figura 51. Listagem de idioma do fichário*

**Lista Completa**

Se você especificar DETAIL(\*FULL) no comando CRTPGM, CRTSRVPGM, UPDPGM ou UPDSRVPGM , a listagem incluirá todos os detalhes fornecidos para DETAIL(\*EXTENDED) mais uma listagem de referência cruzada. A Figura 52 na página 162 mostra um exemplo parcial dos dados adicionais fornecidos.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Criar programa de serviço  Listagem de referência cruzada | | |  |  | 15 |
|  | --------Referências-------- | |
| Identificador | Defs | Ref | Ref | Tipo | Biblioteca | Objeto |
| . | . |  |  | . | . | . |
| . | . |  |  | . | . | . |
| . | . |  |  | . | . | . |
| xlatewt | 000000DD |  |  | \*SRVPGM | \*LIBL | QC2UTIL1 |
| Yn | 00000140 |  |  | \*SRVPGM | \*LIBL | QC2UTIL2 |
| y0 | 0000013E |  |  | \*SRVPGM | \*LIBL | QC2UTIL2 |
| y1 | 0000013F |  |  | \*SRVPGM | \*LIBL | QC2UTIL2 |
| Quantidade | 00000002 |  |  | \*MÓDULO | MYLIB | DINHEIRO |
| Calc1 | 00000011 |  |  | \*MÓDULO | MYLIB | CALCS |
| Calc2 | 00000012 |  |  | \*MÓDULO | MYLIB | CALCS |
| FecharConta | 00000019 |  |  | \*MÓDULO | MYLIB | ACCTS |
| CEECRHP | 000001A0 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEECZST | 0000019F |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEDATE | 000001A9 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEDATM | 000001B1 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEDAYS | 000001A8 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEDBACALHAU | 00000187 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEDSHP | 000001A1 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEDYWK | 000001B3 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEFMDA | 000001AD |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEFMDT | 000001AF |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEFMTM | 000001AE |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEFRST | 0000019E |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEGMT | 000001B6 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEGPID | 00000195 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEGTST | 0000019D |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEISEC | 000001B0 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEELOCT | 000001B4 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEMGET | 00000183 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEMKHP | 000001A2 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEMOUT | 00000184 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEMRCR | 00000182 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEMSG | 00000185 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEENCOD | 00000186 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEQCEN | 000001AC |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEERLHP | 000001A3 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEESCEN | 000001AB |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEESECI | 000001B2 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEESECS | 000001AA |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEESGL | 00000190 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEETREC | 00000191 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEUTC | 000001B5 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEEUTCO | 000001B7 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEE4ABN | 00000192 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEE4CpyDvfb | 0000019A |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEE4CpyIofb | 00000199 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEE4CpyOfb | 00000198 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEE4DAS | 000001A4 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEE4FCB | 0000018A |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEE4HC | 00000197 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEE4RAGE | 0000018B |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| CEE4RIN | 00000196 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| OpenAccount | 00000018 |  |  | \*MÓDULO | MYLIB | ACCTS |
| Pagamento | 00000003 |  |  | \*MÓDULO | MYLIB | DINHEIRO |
| Q LE leBdyCh | 00000188 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| Q LE leBdyEpilog | 00000189 |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| Q LE leDefaultEh | 00000180 | 00000007 | 0000000E | \*SRVPGM | \*LIBL | QLEAWI |
|  | 00000015 |  | 0000001C |  |  |  |
| Q LE mhConversionEh | 00000181 | 00000008 | 0000000F | \*SRVPGM | \*LIBL | QLEAWI |
|  | 00000016 |  | 0000001D |  |  |  |
| Q LE AG\_prod\_rc | 0000017F | 00000004 | 0000000C | \*SRVPGM | \*LIBL | QLEAWI |
|  | 00000013 | 0000001A |  |  |  |  |
| Q LE AG\_user\_rc | 0000017E | 00000005 | 0000000D | \*SRVPGM | \*LIBL | QLEAWI |
|  |  | 00000014 | 0000001B |  |  |  |
| Q LE HdlrRouterEh | 0000018F |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| Q LE RtxRouterCh | 0000018E |  |  | \*SRVPGM | \*LIBL | QLEAWI |
| Taxa | 0000000B |  |  | \*MÓDULO | MYLIB | TAXAS |
| Prazo | 0000000A |  |  | \*MÓDULO | MYLIB | TAXAS |

*Figura 52. Listagem* de *referência cruzada*

**Componentes** de **listagem do** IPA

As seções a seguir descrevem os componentes IPA da listagem:

v Object File Map v Compiler Options Map v Inline Report v Global Symbols Map v Partition Map v Source File Map v Mensagens v Message Summary

O comando CRTPGM ou CRTSRVPGM gera todas essas seções, exceto o relatório embutido, se você especificar IPA(\*YES) e DETAIL(\*BASIC ou \*EXTENDED). O comando CRTPGM ou CRTSRVPGM gera o relatório embutido somente se você especificar IPA(\*YES) e DETAIL(\*FULL).

#### Mapa do arquivo de objeto

A seção de listagem Mapa de Arquivo de Objeto exibe os nomes dos arquivos de objeto que foram usados como entrada para o IPA. Outras seções de listagem, como o Mapa do Arquivo de Origem, usam os números de ID FILE que aparecem nesta seção de listagem.

#### Mapa de opções do compilador

A seção de listagem Mapa de Opções do Compilador identifica as opções do compilador que foram especificadas nos dados IL para cada unidade de compilação processada. Para cada unidade de compilação, ele exibe as opções que são relevantes para o processamento IPA. Você pode especificar essas opções por meio de uma opção de compilador, uma diretiva #pragma ou como valores padrão .

#### Relatório embutido

A seção de listagem Relatório embutido descreve as ações executadas pelo embutido do IPA. Neste relatório, o termo 'subprograma' é equivalente a uma função C/C++ ou a um método C++ . O resumo contém informações como: v Nome de cada subprograma definido. O IPA classifica os nomes dos subprogramas em ordem alfabética.

v Motivo da ação em um subprograma:

* Você especificou #pragma noinline para o subprograma.
* Você especificou #pragma embutido para o subprograma.
* O IPA executou o inlining automático no subprograma.
* Não havia razão para alinhar o subprograma.
* Houve um conflito de partição.
* O IPA não pôde embutir o subprograma porque os dados de IL não existiam. v Ação em um subprograma:
* Subprograma embutido IPA pelo menos uma vez.
* O IPA não in-line subprograma devido a restrições de tamanho iniciais.
* O IPA não in-line subprogramou devido à expansão além da restrição de tamanho.
* O subprograma era um candidato para inlining, mas o IPA não o inlineu.
* Subprograma era candidato a inlining, mas não foi referido.
* O subprograma é diretamente recursivo, ou algumas chamadas têm parâmetros incompatíveis.

v Status do subprograma original após a inserção:

* O IPA descartou o subprograma porque ele não é mais referido e é definido como interno estático.
* O IPA não descartou o subprograma, por vários motivos:
  + O subprograma é externo. (Ele pode ser chamado de fora da unidade de compilação.) - A chamada de subprograma para este subprograma permanece.
  + O subprograma tem seu endereço tomado.

v Tamanho relativo inicial do subprograma (em Abstract Code Units). v Tamanho relativo final do subprograma (em Abstract Code Units) após a inlinagem.

v O número de chamadas dentro do subprograma e o número dessas chamadas que o IPA inseriu no

subprograma.

v O número de vezes que o subprograma é chamado por outros na unidade de compilação e o número de vezes que o IPA alinhou o subprograma.

v O modo selecionado e o valor de limite e limite especificados. Funções estáticas cujos nomes podem não ser exclusivos dentro do aplicativo como um todo terão nomes prefixados com @nnn@ ou XXXX@nnn@, onde XXXX é o nome da partição e onde nnn é o número do arquivo de origem.

A estrutura detalhada da chamada contém informações específicas de cada subprograma, como:

v Subprogramas que ele chama. v Subprogramas que o chamam. v Subprogramas nos quais ele está embutido.

As informações podem permitir uma melhor análise do programa se você quiser usar o inliner no modo seletivo. As contagens neste relatório não incluem chamadas de programas não-IPA para IPA.

#### Mapa de símbolos globais

A seção de listagem Mapa de Símbolos Globais mostra como os símbolos globais são mapeados em membros de estruturas de dados globais pelo processo de otimização de coalescência de variáveis globais . Ele inclui informações de símbolo e informações de nome de arquivo ( as informações de nome de arquivo podem ser aproximadas). Além disso, informações sobre o número da linha podem estar disponíveis.

#### Mapa de Partição

A seção de listagem Mapa de Partição descreve cada uma das partições de código objeto criadas pelo IPA. Ele fornece as seguintes informações:

v O motivo para gerar cada partição. v As opções usadas para gerar o código objeto. v A função e os dados globais incluídos na partição . v Os arquivos de origem que foram usados para criar a partição.

**Mapa do arquivo de origem**

A seção de listagem Mapa de Arquivos de Origem identifica os arquivos de origem incluídos nos arquivos de objeto.

#### Mensagens

Se o IPA detectar um erro ou a possibilidade de um erro, ele emitirá uma ou mais mensagens de diagnóstico e gerará a seção Listagem de mensagens. Esta seção de listagem contém um resumo das mensagens emitidas durante o processamento do IPA. As mensagens são classificadas por gravidade. A seção de listagem Mensagens exibe o número da página de listagem onde cada mensagem foi originalmente mostrada. Ele também exibe o texto da mensagem e, opcionalmente, informações relacionadas a um nome de arquivo, linha (se conhecido) e coluna (se conhecido).

#### Resumo da mensagem

A seção de listagem Resumo da Mensagem exibe o número total de mensagens e o número de mensagens para cada nível de gravidade.

**Listando, por exemplo,** **o programa de serviço**

A Figura 48 na página 159, a Figura 50 na página 160 e a Figura 52 na página 162 mostram alguns dos dados de listagem gerados quando DETAIL(\*FULL) foi especificado para criar o programa de serviço FINANCEIRO na Figura 35 na página 75. As figuras mostram as estatísticas de vinculação, a listagem de informações do fichário e a listagem de referência cruzada.

#### Listagem de informações do fichário para o programa de serviço de exemplo

A listagem de informações do fichário (Figura 50 na página 160) inclui os seguintes dados e cabeçalhos de coluna: v A biblioteca e o nome do módulo ou programa de serviço que foi processado.

Se o campo Acoplado mostrar um valor de \*SIM para um objeto de módulo, o módulo será marcado para ser acoplado por cópia. Se o campo *Acoplado* mostrar um valor de \*SIM para um programa de serviço, o programa de serviço será vinculado por referência. Se o campo *Acoplado* mostrar um valor de \*NO para um objeto de módulo ou programa de serviço, esse objeto não será incluído na associação. O motivo é que o objeto não forneceu uma exportação que satisfizesse uma importação não resolvida.

v Número

Para cada módulo ou programa de serviço que foi processado, um identificador exclusivo (ID) é associado a cada exportação (definição) ou importação (referência).

v Símbolo

Esta coluna identifica o nome do símbolo como uma exportação (Def) ou uma importação (Ref). v Ref

Um número especificado nesta coluna (Ref) é o ID exclusivo da exportação (Def) que satisfaz a solicitação de importação. Por exemplo, na Figura 50 na página 160, a ID exclusiva para o 00000005 de importação corresponde à ID exclusiva para a exportação 0000017E.

v Identificador

Este é o nome do símbolo que é exportado ou importado. O nome do símbolo importado para o 00000005 de ID exclusivo é Q LE AG\_user\_rc. O nome do símbolo exportado para o ID exclusivo 0000017E também é Q LE AG\_user\_rc. v Type

Se o nome do símbolo for um procedimento, ele será identificado como Proc. Se o nome do símbolo for um item de dados, ele será identificado como Dados.

v Âmbito de aplicação

Para módulos, esta coluna identifica se um nome de símbolo exportado é acessado no nível do módulo ou na interface pública de um programa de serviço. Se um programa estiver sendo criado, os nomes de símbolos exportados poderão ser acessados somente no nível do módulo. Se um programa de serviço estiver sendo criado, os nomes de símbolos exportados poderão ser acessados no nível do módulo ou no nível do programa de serviço (SrvPgm). Se um símbolo exportado fizer parte da interface pública, o valor na coluna *Escopo* deverá ser SrvPgm. v Export

Esta coluna identifica a força de um item de dados que é exportado de um módulo ou programa de serviço. v Chave

Esta coluna contém informações adicionais sobre quaisquer exportações fracas. Normalmente, esta coluna está em branco.

#### Listagem de referência cruzada para o programa de serviço de exemplo

A listagem de referência cruzada na Figura 52 na página 162 é outra maneira de examinar os dados apresentados nas informações do fichário. A listagem de referência cruzada inclui os seguintes títulos de coluna:

v Identificador

O nome da exportação que foi processada durante a resolução do símbolo. v Defs

O ID exclusivo associado a cada exportação. v Refs

Um número nesta coluna indica a ID exclusiva da importação (Ref) que foi resolvida para esta exportação (Def).

v Tipo

Identifica se a exportação veio de um \*MODULE ou de um objeto \*SRVPGM.

v Biblioteca

O nome da biblioteca conforme especificado no comando ou no diretório de vinculação.

v Objeto

O nome do objeto que forneceu a exportação (Def).

#### Estatísticas de Vinculação, por exemplo, Programa de Serviço

A Figura 48 na página 159 mostra um conjunto de estatísticas para a criação do programa de serviço FINANCEIRO. As estatísticas identificam onde o fichário passou algum tempo quando estava processando a solicitação de criação. Você tem apenas controle indireto sobre os dados apresentados nesta seção. Alguma quantidade de sobrecarga de processamento não pode ser medida. Portanto, o valor listado no campo *Tempo total da CPU*  é maior do que a soma dos tempos listados nos campos anteriores.

**Erros de idioma do fichário**

Enquanto o sistema está processando a linguagem do fichário durante a criação de um programa de serviço, um erro pode ocorrer. Se DETAIL(\*EXTENDED) ou DETAIL(\*FULL) for especificado no comando Create Service Program (CRTSRVPGM), você poderá ver os erros no arquivo spooled.

As seguintes mensagens informativas podem ocorrer:

v Assinatura acolchoada v Assinatura truncada

Os seguintes erros de aviso podem ocorrer: v Interface de limites de bloco de exportação atual v Bloco de exportação duplicado v Símbolo duplicado na exportação anterior v A verificação de nível não pode ser desabilitada mais do que uma vez, ignorado v Vários blocos de exportação atuais não permitidos, anteriormente assumido

Os seguintes erros graves podem ocorrer:

v O bloco de exportação atual está vazio

v Bloco de exportação não concluído, fim do arquivo encontrado antes do ENDPGMEXP v Bloco de exportação não iniciado, STRPGMEXP necessário v Blocos de exportação não podem ser aninhados, ENDPGMEXP ausente v Exportações devem existir dentro de blocos de exportação v Assinaturas idênticas para blocos de exportação diferentes, devem alterar exportações v Várias correspondências curinga v Nenhum bloco de exportação atual v Nenhuma correspondência curinga v Bloco de exportação anterior é vazio v Assinatura contém caracteres variantes v ASSINATURA(\*GEN) necessários com LVLCHK(\*NO) v Sintaxe de assinatura não válida v Nome do símbolo necessário v Símbolo não permitido como exportação do programa de serviço

v Símbolo não definido v Sintaxe não válida

**Assinatura acolchoada**

A Figura 53 mostra uma listagem de idioma do fichário que contém essa mensagem.

Listagem de idioma do fichário

STRPGMEXP SIGNATURE('Assinatura curta')

Assinatura acolchoada

SÍMBOLO DE EXPORTAÇÃO('Proc\_2') ENDPGMEXP

Assinatura de exportação: E2889699A340A289879581A3A4998540.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 53. A assinatura fornecida era menor que 16 bytes, por isso é acolchoada*

Esta é uma mensagem informativa.

#### Alterações sugeridas

Nenhuma alteração é necessária.

Se você deseja evitar a mensagem, certifique-se de que a assinatura que está sendo fornecida é exatamente 16 bytes de comprimento.

**Assinatura truncada**

A Figura 54 mostra uma listagem de idioma do fichário que contém essa mensagem.

Listagem de idioma do fichário

STRPGMEXP SIGNATURE('Esta assinatura é muito longa')

Assinatura truncada

SÍMBOLO DE EXPORTAÇÃO('Proc\_2') ENDPGMEXP

Assinatura de exportação: E38889A240A289879581A3A499854089.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 54. Somente os primeiros 16 bytes de dados fornecidos são usados para a assinatura*

Esta é uma mensagem informativa.

#### Alterações sugeridas

Nenhuma alteração é necessária.

Se você deseja evitar a mensagem, certifique-se de que a assinatura que está sendo fornecida é exatamente 16 bytes de comprimento.

**Interface atual de limites de bloco de exportação**

A Figura 55 na página 168 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

STRPGMEXP PGMLVL(\*ATUAL)

SÍMBOLO DE EXPORTAÇÃO (A)

SÍMBOLO DE EXPORTAÇÃO (B)

ENDPGMEXP

Assinatura de exportação: 00000000000000000000000000000000CD2.

STRPGMEXP PGMLVL(\*PRV)

SÍMBOLO DE EXPORTAÇÃO (A)

SÍMBOLO DE EXPORTAÇÃO (B)

SÍMBOLO DE EXPORTAÇÃO (C)

ENDPGMEXP

Assinatura de exportação: 0000000000000000000000000000000CDE3. \*\*\*\* Interface atual de limites de bloco de exportação.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 55. A PGMLVL(\*PRV) EXPORTOU*  MAIS *SÍMBOLOS DO QUE O PGMLVL(\*CURRENT)*

Este é um erro de aviso.

Um bloco de exportação pgmlvl(\*prv) especificou mais símbolos do que o bloco de exportação PGMLVL(\*CURRENT).

Se nenhum outro erro ocorrer, o programa de serviço será criado.

Se ambas as opções a seguir forem verdadeiras:

v PGMLVL(\*PRV) tinha suportado um procedimento chamado C v Sob o novo programa de serviço, o procedimento C não é mais suportado

qualquer programa ILE ou programa de serviço que chamou o procedimento C neste programa de serviço obtém um erro em tempo de execução.

#### Alterações sugeridas

1. Verifique se o bloco de exportação PGMLVL(\*CURRENT) tem mais símbolos a serem exportados do que um bloco de exportação PGMLVL(\*PRV).
2. Execute o comando CRTSRVPGM novamente.

Neste exemplo, o EXPORT SYMBOL(C ) foi adicionado incorretamente ao bloco STRPGMEXP PGMLVL(\*PRV) em vez do bloco PGMLVL(\*CURRENT).

**Bloco de exportação duplicado**

A Figura 56 na página 169 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

STRPGMEXP PGMLVL(\*ATUAL)

SÍMBOLO DE EXPORTAÇÃO (A)

SÍMBOLO DE EXPORTAÇÃO (B)

ENDPGMEXP

Assinatura de exportação: 00000000000000000000000000000000CD2.

STRPGMEXP PGMLVL(\*PRV)

SÍMBOLO DE EXPORTAÇÃO (A)

SÍMBOLO DE EXPORTAÇÃO (B)

ENDPGMEXP

Assinatura de exportação: 000000000000000000000000000000000CD2. \*\*\*\*\*\*\*\* Bloco de exportação duplicado.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 56. Blocos STRPGMEXP/ENDPGMEXP duplicados*

Este é um erro de aviso.

Mais de um bloco STRPGMEXP e ENDPGMEXP exportou todos os mesmos símbolos na mesma ordem.

Se nenhum outro erro ocorrer, o programa de serviço será criado. A assinatura duplicada é incluída apenas uma vez no programa de serviço criado.

#### Alterações sugeridas

1. Faça uma das seguintes alterações: v Verifique se o bloco de exportação PGMLVL(\*CURRENT) está correto. Atualize-o conforme apropriado.

v Remova o bloco de exportação duplicado.

1. Execute o comando CRTSRVPGM novamente.

Neste exemplo, o comando STRPGMEXP com PGMLVL(\*CURRENT) especificado precisa ter a seguinte linha de origem adicionada após EXPORT SYMBOL(B):

SÍMBOLO DE EXPORTAÇÃO (C)

**Símbolo duplicado na exportação anterior**

A Figura 57 mostra uma listagem de idioma do fichário que contém um erro de símbolo duplicado.

Listagem de idioma do fichário

STRPGMEXP PGMLVL(\*ATUAL)

SÍMBOLO DE EXPORTAÇÃO (A)

SÍMBOLO DE EXPORTAÇÃO (B)

SÍMBOLO DE EXPORTAÇÃO (A)

Símbolo duplicado na exportação anterior

SÍMBOLO DE EXPORTAÇÃO (C)

ENDPGMEXP

Assinatura de exportação: 00000000000000000000000000000CDED3.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 57. Duplicar símbolos exportados*

Este é um erro de aviso.

Um símbolo a ser exportado do programa de serviço foi especificado mais de uma vez em um bloco STRPGMEXP e ENDPGMEXP.

Se nenhum outro erro ocorrer, o programa de serviço será criado. Somente o primeiro símbolo duplicado é exportado do programa de serviço. Todos os símbolos duplicados afetam a assinatura que é gerada.

#### Alterações sugeridas

1. Remova uma das linhas de origem duplicadas do arquivo de origem do idioma do fichário.
2. Execute o comando CRTSRVPGM novamente.

Neste exemplo, remova o segundo EXPORT SYMBOL(A).

**A verificação** de  **nível não pode ser desabilitada mais de uma vez, ignorada**

A Figura 58 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

STRPGMEXP PGMLVL(\*CURRENT) LVLCHK(\*NÃO)

SÍMBOLO DE EXPORTAÇÃO (A)

SÍMBOLO DE EXPORTAÇÃO (B)

ENDPGMEXP

Assinatura de exportação: 0000000000000000000000000000000000000000.

STRPGMEXP PGMLVL(\*PRV) LVLCHK(\*NÃO)

A verificação de nível não pode ser desabilitada mais de uma vez, ignorado EXPORT SYMBOL(A)

ENDPGMEXP

Assinatura de exportação: 00000000000000000000000000000000C1.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 58. Vários* *comandos STRPGMEXP têm LVLCHK(\*NO) especificado*

Este é um erro de aviso.

Mais de um STRPGMEXP blocos especificados LVLCHK(\*NO).

Se nenhum outro erro ocorrer, o programa de serviço será criado. O segundo e subsequente LVLCHK(\*NO) são assumidos como LVLCHK(\*YES).

#### Alterações sugeridas

1. Certifique-se de que apenas um bloco STRPGMEXP tenha LVLCHK(\*NO) especificado.
2. Execute o comando CRTSRVPGM novamente.

Neste exemplo, o bloco de exportação PGMLVL(\*PRV) é o único bloco de exportação que tem LVLCHK(\*NO) especificado. O valor LVLCHK(\*NO) é removido do bloco de exportação PGMLVL(\*CURRENT).

**Vários blocos de exportação atuais não permitidos, assumidos anteriormente**

A Figura 59 na página 171 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

|  |  |
| --- | --- |
| STRPGMEXP PGMLVL(\*ATUAL)  SÍMBOLO DE EXPORTAÇÃO (A)  SÍMBOLO DE EXPORTAÇÃO (B)  SÍMBOLO DE EXPORTAÇÃO (C)  ENDPGMEXP  Assinatura de exportação: 000000000000000000000000000000CDE3. STRPGMEXP  SÍMBOLO DE EXPORTAÇÃO (A)  Vários blocos de exportação "atuais" não são permitidos, "anteriores" assumidos. SÍMBOLO DE EXPORTAÇÃO (B)  ENDPGMEXP  Assinatura de exportação: 00000000000000000000000000000000CD2. |  |
| \* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G | \* \* \* \* \* |

*Figura 59. Mais de um valor PGMLVL(\*CURRENT) especificado*

Este é um erro de aviso.

Um valor de PGMLVL(\*CURRENT) foi especificado ou foi permitido como padrão PGMLVL(\*CURRENT) em mais de um comando STRPGMEXP. O segundo e subsequentes blocos de exportação com um valor de PGMLVL(\*CURRENT) são considerados PGMLVL(\* PRV).

Se nenhum outro erro ocorrer, o programa de serviço será criado.

#### Alterações sugeridas

1. Altere o texto de origem apropriado para STRPGMEXP PGMLVL(\*PRV).
2. Execute o comando CRTSRVPGM novamente.

Neste exemplo, o segundo STRPGMEXP é o único a mudar.

**Bloco de exportação** atual **está vazio**

A Figura 60 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

STRPGMEXP PGMLVL(\*ATUAL)

ENDPGMEXP

Assinatura de exportação: 000000000000000000000000000000000000. \*\*\*ERRO O bloco de exportação atual está vazio.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 60. Nenhum símbolo a ser exportado do bloco STRPGMEXP PGMLVL(\*CURRENT)*

Trata-se de um erro grave.

Nenhum símbolo é identificado para ser exportado do bloco de exportação \*ATUAL.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Faça uma das seguintes alterações:

v Adicione os nomes dos símbolos a serem exportados.

v Remova o bloco STRPGMEXP-ENDPGMEXP vazio e faça outro bloco STRPGMEXPENDPGMEXP como PGMLVL(\*CURRENT).

1. Execute o comando CRTSRVPGM.

Neste exemplo, a seguinte linha de origem é adicionada ao arquivo de origem do idioma do fichário entre o

Comandos STRPGMEXP e ENDPGMEXP:

SÍMBOLO DE EXPORTAÇÃO (A)

**Bloco**  de exportação **não concluído, fim de arquivo encontrado antes** do  **ENDPGMEXP**

A Figura 61 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

STRPGMEXP PGMLVL(\*CURRENT) \*\*\*ERRO Sintaxe não válida.

ERRO Bloco de exportação não concluído, fim do arquivo encontrado antes do ENDPGMEXP.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 61. Nenhum comando ENDPGMEXP encontrado, mas o fim do arquivo de origem foi encontrado*

Trata-se de um erro grave.

Nenhum ENDPGMEXP foi encontrado antes que o final do arquivo fosse atingido.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Faça uma das seguintes alterações:

v Adicione o comando ENDPGMEXP no local apropriado.

v Remova qualquer comando STRPGMEXP que não tenha um comando ENDPGMEXP correspondente e remova todos os nomes de símbolos a serem exportados.

1. Execute o comando CRTSRVPGM.

Neste exemplo, as seguintes linhas são adicionadas após o comando STRPGMEXP:

SÍMBOLO DE EXPORTAÇÃO (A)

ENDPGMEXP

**Bloco de**  exportação **não iniciado, STRPGMEXP necessário**

A Figura 62 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

|  |  |  |
| --- | --- | --- |
| ENDPGMEXP  ERRO Bloco de exportação não iniciado, STRPGMEXP necessário. \*\*\*ERRO Nenhum bloco de exportação 'atual' |  |  |
| \* \* \* \* \*E N D O F B I N D E R L A N G U A G E | L I S T I N G | \* \* \* \* \* |

*Figura 62. Comando STRPGMEXP está ausente*

Trata-se de um erro grave.

Nenhum comando STRPGMEXP foi encontrado antes de encontrar um comando ENDPGMEXP.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Faça uma das seguintes alterações: v Adicione o comando STRPGMEXP.

v Remova todos os símbolos exportados e o comando ENDPGMEXP.

1. Execute o comando CRTSRVPGM.

Neste exemplo, as duas linhas de origem a seguir são adicionadas ao arquivo de origem do idioma do fichário antes do

Comando ENDPGMEXP.

STRPGMEXP

SÍMBOLO DE EXPORTAÇÃO (A)

**Blocos de** exportação  **não podem ser aninhados, ENDPGMEXP ausente**

A Figura 63 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

STRPGMEXP PGMLVL(\*ATUAL)

SÍMBOLO DE EXPORTAÇÃO (A)

SÍMBOLO DE EXPORTAÇÃO (B)

STRPGMEXP PGMLVL(\*PRV)

ERRO Os blocos de exportação não podem ser aninhados, faltando ENDPGMEXP. SÍMBOLO DE EXPORTAÇÃO (A)

ENDPGMEXP

Assinatura de exportação: 00000000000000000000000000000000C1.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 63. Comando ENDPGMEXP está ausente*

Trata-se de um erro grave.

Nenhum comando ENDPGMEXP foi encontrado antes de encontrar outro comando STRPGMEXP.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Faça uma das seguintes alterações:

v Adicione o comando ENDPGMEXP antes do próximo comando STRPGMEXP. v Remova o comando STRPGMEXP e quaisquer nomes de símbolos a serem exportados.

1. Execute o comando CRTSRVPGM.

Neste exemplo, um comando ENDPGMEXP é adicionado ao arquivo de origem do fichário antes do segundo comando STRPGMEXP.

**As exportações devem existir dentro dos blocos de exportação**

A Figura 64 na página 174 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

STRPGMEXP PGMLVL(\*ATUAL)

SÍMBOLO DE EXPORTAÇÃO (A)

SÍMBOLO DE EXPORTAÇÃO (B)

ENDPGMEXP

Assinatura de exportação: 00000000000000000000000000000000CD2. SÍMBOLO DE EXPORTAÇÃO (A)

ERRO As exportações devem existir dentro dos blocos de exportação.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 64. Nome* do símbolo  *a ser exportado está fora do bloco STRPGMEXP-ENDPGMEXP* Este é um erro grave.

Um símbolo a ser exportado não é definido dentro de um bloco STRPGMEXP-ENDPGMEXP.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Faça uma das seguintes alterações:

v Mova o símbolo a ser exportado. Coloque-o dentro de um bloco STRPGMEXP-ENDPGMEXP.

v Remova o símbolo.

1. Execute o comando CRTSRVPGM.

Neste exemplo, a linha de origem em erro é removida do arquivo de origem do idioma do fichário .

**Assinaturas idênticas para blocos de exportação diferentes, devem alterar as exportações**

Trata-se de um erro grave.

Assinaturas idênticas foram geradas a partir de blocos STRPGMEXP-ENDPGMEXP que exportaram símbolos diferentes. É altamente improvável que essa condição de erro ocorra. Para qualquer conjunto de símbolos não triviais a serem exportados, esse erro deve ocorrer apenas uma vez a cada tentativa 3.4E28.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Faça uma das seguintes alterações:

v Adicione um símbolo adicional a ser exportado do bloco PGMLVL(\*CURRENT).

O método preferencial é especificar um símbolo que já está exportado. Isso causaria um erro de aviso de símbolos duplicados, mas ajudaria a garantir que uma assinatura seja exclusiva. Um método alternativo é adicionar outro símbolo a ser exportado que não tenha sido exportado.

v Altere o nome de um símbolo a ser exportado de um módulo e faça a alteração correspondente no arquivo de origem do idioma do fichário.

v Especifique uma assinatura usando o parâmetro SIGNATURE no comando Iniciar exportação de programa (STRPGMEXP).

1. Execute o comando CRTSRVPGM.

**Várias correspondências curinga**

A Figura 65 na página 175 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

STRPGMEXP PGMLVL(\*ATUAL)

EXPORTAÇÃO ("A"<<<)

ERRO Várias correspondências da especificação curinga

EXPORTAÇÃO ("B"<<<)

ENDPGMEXP

Assinatura de exportação: 000000000000000000000000000000000FFC2.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G

*Figura 65. Várias correspondências de especificação* *curinga*

Trata-se de um erro grave.

Um curinga especificado para exportação correspondia a mais de um símbolo disponível para exportação.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Especifique um curinga com mais detalhes para que a exportação correspondente desejada seja a única exportação correspondente.
2. Execute o comando CRTSRVPGM.

**Nenhum bloco de exportação**  atual

A Figura 66 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

STRPGMEXP PGMLVL(\*PRV) SÍMBOLO DE EXPORTAÇÃO(A)

ENDPGMEXP

Assinatura de exportação: 00000000000000000000000000000000C1.

ERRO Nenhum bloco de exportação 'atual'

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 66. Nenhum bloco* de *exportação* PGMLVL(\*CURRENT)

Trata-se de um erro grave.

Nenhum STRPGMEXP PGMLVL(\*CURRENT) é encontrado no arquivo de origem do idioma do fichário.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Faça uma das seguintes alterações:

v Alterar um PGMLVL(\*PRV) para PGMLVL(\* CURRENT).

v Adicione um bloco STRPGMEXP-ENDPGMEXP que seja o bloco de exportação \*CURRENT correto.

1. Execute o comando CRTSRVPGM.

Neste exemplo, o PGMLVL(\*PRV) é alterado para PGMLVL(\*CURRENT).

**Sem correspondências curinga**

A Figura 67 na página 176 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

STRPGMEXP PGMLVL(\*ATUAL)

EXPORTAÇÃO ("Z"<<<)

ERRO Nenhuma correspondência da especificação curinga

EXPORTAÇÃO ("B"<<<)

ENDPGMEXP

Assinatura de exportação: 000000000000000000000000000000000FFC2.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G

*Figura 67. Nenhuma correspondência de especificação* *curinga*

Trata-se de um erro grave.

Um curinga especificado para exportação não correspondia a nenhum símbolo disponível para exportação.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Especifique um curinga que corresponda ao símbolo desejado para exportação.
2. Execute o comando CRTSRVPGM.

**Bloco de exportação** anterior **está vazio**

A Figura 68 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

STRPGMEXP PGMLVL(\*ATUAL)

SÍMBOLO DE EXPORTAÇÃO (A)

SÍMBOLO DE EXPORTAÇÃO (B)

ENDPGMEXP

Assinatura de exportação: 00000000000000000000000000000000CD2.

STRPGMEXP PGMLVL(\*PRV)

ENDPGMEXP

Assinatura de exportação: 00000000000000000000000000000000000. \*\*\*ERRO O bloco de exportação anterior está vazio.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 68. Nenhum bloco* de *exportação* PGMLVL(\*CURRENT)

Trata-se de um erro grave.

Um strpgmexp pgmlvl(\*prv ) foi encontrado, e nenhum símbolo foi especificado.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Faça uma das seguintes alterações:

v Adicione símbolos ao bloco STRPGMEXP-ENDPGMEXP que está vazio.

v Remova o bloco STRPGMEXP-ENDPGMEXP que está vazio.

1. Execute o comando CRTSRVPGM.

Neste exemplo, o bloco STRPGMEXP-ENDPGMEXP vazio é removido do arquivo de origem do idioma do fichário.

**A assinatura contém** **caracteres variantes**

A Figura 69 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

ASSINATURA STRPGMEXP('\! cdefghijklmnop')

A assinatura ERROR contém caracteres variantes

SÍMBOLO DE EXPORTAÇÃO('Proc\_2') ENDPGMEXP

Assinatura de exportação: E05A8384858687888991929394959697.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 69. A assinatura contém* *caracteres variantes*

Trata-se de um erro grave.

A assinatura contém caracteres que não estão em todos os identificadores de conjunto de caracteres codificados (CCSIDs).

O programa de serviço não é criado.

#### Alterações sugeridas

1. Remova os caracteres variantes.
2. Execute o comando CRTSRVPGM.

Neste caso específico, é o \! que precisa ser removido.

**ASSINATURA(\*GEN) Necessária com LVLCHK(\*NO)**

A Figura 70 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

ASSINATURA STRPGMEXP('ABCDEFGHIJKLMNOP') LVLCHK(\*NÃO)

SÍMBOLO DE EXPORTAÇÃO(«Proc\_2»)

ASSINATURA DE ERRO(\*GEN) necessária com LVLCHK(\*NO)

ENDPGMEXP

Assinatura de exportação: C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 70. Se LVLCHK(\*NO) for especificado, uma assinatura explícita não* é  *válida* Este é um erro grave.

Se LVLCHK(\*NO) for especificado, SIGNATURE(\*GEN) será necessário.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Faça uma das seguintes alterações: v Especificar SIGNATURE(\*GEN) v Especificar LVLCHK(\*YES)
2. Execute o comando CRTSRVPGM.

**Sintaxe de** assinatura  **não válida**

A Figura 71 mostra uma listagem de idioma do fichário que contém esse erro.

|  |  |  |
| --- | --- | --- |
| Listagem de idioma do fichário  ASSINATURA DE STRPFMEX('"ABCDEFGHIJKL"')  ERRO Sintaxe de assinatura não válida  ERRO Sintaxe de assinatura não válida  ERRO Sintaxe não válida.  ERRO Sintaxe não válida.  SÍMBOLO DE EXPORTAÇÃO('Proc\_2') ENDPGMEXP  \* \* \* \* \*E N D O F B I N D E R L A N G U A G E  *Figura 71. O que* é  *especificado para o valor de assinatura não é válido*  Trata-se de um erro grave.  A assinatura contém caracteres que não são válidos.  O programa de serviço não é criado.  **Alterações sugeridas** | L I S T I N G | \* \* \* \* \* |
| 1. Remova os caracteres que não são válidos do valor de assinatura. 2. Execute o comando CRTSRVPGM.   Nesse caso, remova os caracteres " do campo de assinatura.  **Nome do** símbolo  **necessário**  A Figura 72 mostra uma listagem de idioma do fichário que contém esse erro.  Listagem de idioma do fichário  STRPGMEXP PGMLVL(\*ATUAL)  SÍMBOLO DE EXPORTAÇÃO (A)  SÍMBOLO DE EXPORTAÇÃO (')  ERRO Nome do símbolo necessário.  ENDPGMEXP  Assinatura de exportação: 00000000000000000000000000000000C1. | |
| \* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G | | \* \* \* \* \* |

*Figura 72. Nenhum símbolo a ser exportado*

Trata-se de um erro grave.

Nenhum nome de símbolo foi encontrado para exportar do programa de serviço.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Faça uma das seguintes alterações:

v Remova a linha de erro do arquivo de origem do idioma do fichário. v Adicione um nome de símbolo a ser exportado do programa de serviço.

1. Execute o comando CRTSRVPGM.

Neste exemplo, a linha de origem EXPORT SYMBOL("") é removida do arquivo de origem do idioma do fichário .

**Símbolo não permitido como exportação** do **programa de** serviço

A Figura 73 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

STRPGMEXP PGMLVL(\*ATUAL)

SÍMBOLO DE EXPORTAÇÃO (A)

Símbolo de erro não permitido como exportação do programa de serviço.

SÍMBOLO DE EXPORTAÇÃO(D)

ENDPGMEXP

Assinatura de exportação: 000000000000000000000000000000000CD4.

\* \* \* \* \*E N D O F B I N D E R L A N G U A G E L I S T I N G\* \* \* \* \* \*

*Figura 73. Nome* do símbolo  *não válido para exportar do programa de serviço*

Trata-se de um erro grave.

O símbolo a ser exportado do programa de serviço não foi exportado de um dos módulos a serem vinculados por cópia. Normalmente, o símbolo especificado para ser exportado do programa de serviço é, na verdade, um símbolo que precisa ser importado pelo programa de serviço.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Faça uma das seguintes alterações:

v Remova o símbolo com erro do arquivo de origem do idioma do fichário.

v No parâmetro MODULE do comando CRTSRVPGM, especifique o módulo que tem o símbolo desejado a ser exportado.

v Adicione o símbolo a um dos módulos que serão vinculados por cópia e recrie o objeto de módulo.

1. Execute o comando CRTSRVPGM.

Neste exemplo, a linha de origem de EXPORT SYMBOL(A) é removida do arquivo de origem do idioma do fichário .

**Símbolo não definido**

A Figura 74 mostra uma listagem de idioma do fichário que contém esse erro.

Listagem de idioma do fichário

STRPGMEXP PGMLVL(\*ATUAL)

SÍMBOLO DE EXPORTAÇÃO (A)

SÍMBOLO DE EXPORTAÇÃO (Q)

|  |  |  |
| --- | --- | --- |
| Símbolo de erro não definido.  ENDPGMEXP  Assinatura da exportação: 00000000000000000000000000000000CE8. |  |  |
| \* \* \* \* \*E N D O F B I N D E R L A N G U A G E | L I S T I N G | \* \* \* \* \* |

*Figura 74. Símbolo não encontrado nos módulos que* *devem ser vinculados por cópia*

Trata-se de um erro grave.

O símbolo a ser exportado do programa de serviço não pôde ser encontrado nos módulos que devem ser vinculados por cópia.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Faça uma das seguintes alterações:

v Remova o símbolo que não está definido do arquivo de origem do idioma do fichário.

v No parâmetro MODULE do comando CRTSRVPGM, especifique o módulo que tem o símbolo desejado a ser exportado.

v Adicione o símbolo a um dos módulos que serão vinculados por cópia e recrie o objeto de módulo.

1. Execute o comando CRTSRVPGM.

Neste exemplo, a linha de origem de EXPORT SYMBOL(Q) é removida do arquivo de origem do idioma do fichário .

**Sintaxe não válida**

Trata-se de um erro grave.

As instruções no membro de origem não são instruções de idioma de fichário válidas.

O programa de serviço não é criado.

#### Alterações sugeridas

1. Corrija o membro de origem para que ele contenha instruções de idioma do fichário válidas.
2. Execute o comando CRTSRVPGM.

**Apêndice B.**  **Exceções em programas** **otimizados**

Em circunstâncias raras, uma mensagem de exceção MCH3601 pode ocorrer em programas compilados com nível de otimização 30 (\*FULL) ou 40. Este apêndice explica um exemplo em que essa mensagem ocorre. O mesmo programa não recebe uma mensagem de exceção MCH3601 quando compilado com nível de otimização 10 (\*NONE) ou 20 (\* BASIC). Se a mensagem neste exemplo ocorrerá depende de como o compilador ILE HLL aloca o armazenamento para matrizes. Este exemplo pode nunca ocorrer para o seu idioma.

Quando você solicita o nível de otimização 30 (\*FULL) ou 40, o ILE tenta melhorar o desempenho calculando as referências de índice de matriz fora dos loops. Quando você se refere a uma matriz em um loop, muitas vezes você está acessando todos os elementos em ordem. O desempenho pode ser melhorado salvando o último endereço do elemento de matriz da iteração de loop anterior. Para realizar essa melhoria de desempenho, o ILE calcula o primeiro endereço do elemento de matriz fora do loop e salva o valor para uso dentro do loop.

Veja o seguinte exemplo:

DCL ARR[1000] INTEIRO;

DCL I INTEIRO;

I = init\_expression; /\* Suponha que init\_expression avalia

para -1 que é então atribuído a I \*/

/\* Mais declarações \*/

WHILE ( Eu < limit\_expression )

I = I + 1;

/\* Algumas instruções no loop while \*/

ARR[I] = some\_expression;

/\* Outras instruções no loop while \*/

FIM;

Se uma referência a ARR[init\_expression] tivesse produzido um índice de matriz incorreto, este exemplo pode causar uma exceção MCH3601. Isso ocorre porque o ILE tentou calcular o primeiro endereço do elemento de matriz antes de inserir o loop WHILE.

Se você receber exceções MCH3601 no nível de otimização 30 (\*FULL) ou 40, procure a seguinte situação:

1. Você tem um loop que incrementa uma variável antes que ela use a variável como um índice de elemento de matriz.
2. O valor inicial da variável de índice na entrada do loop é negativo.
3. Uma referência à matriz usando o valor inicial da variável não é válida.

Quando essas condições existirem, pode ser possível fazer o seguinte para que o nível de otimização 30 (\*FULL) ou 40 ainda possa ser usado:

1. Mova a parte do programa que incrementa a variável para a parte inferior do loop.
2. Altere as referências às variáveis conforme necessário.

O exemplo anterior seria alterado da seguinte forma:

I = init\_expression + 1;

ENQUANTO ( Eu < limit\_expression + 1 )

© Direitos autorais IBM Corp. 1997, 2010

ARR[I] = some\_expression;

I = I + 1;

FIM;

Se essa alteração não for possível, reduza o nível de otimização de 30 (\*FULL) ou 40 para 20 (\*BASIC) ou 10 (\*NENHUM).

**Apêndice C. Comandos** CL  **usados com** **objetos ILE**

As tabelas a seguir indicam quais comandos CL podem ser usados com cada objeto ILE.

**Comandos** CL  **usados com módulos**

*Tabela 14. Comandos* CL  *usados com módulos*

|  |  |
| --- | --- |
| **Comando** | **Nome** Descritivo |
| CHGMOD | Alterar módulo |
| CRTCBLMOD | Criar módulo COBOL |
| CRTCLMOD | Criar módulo CL |
| CRTCMOD | Criar módulo C |
| CRTCPPMOD | Criar módulo C++ |
| CRTRPGMOD | Criar módulo RPG |
| DLTMOD | Excluir módulo |
| DSPMOD | Módulo de exibição |
| RTVBNDSRC | Recuperar origem do fichário |
| WRKMOD | Trabalhar com módulo |

**Comandos** CL  **usados com objetos de programa**

*Tabela 15. Comandos* CL  *usados com objetos de programa*

|  |  |
| --- | --- |
| **Comando** | **Nome** Descritivo |
| CHGPGM | Programa de Mudança |
| CRTBNDC | Criar programa C acoplado |
| CRTBNDCBL | Criar programa COBOL vinculado |
| CRTBNDCL | Criar programa CL acoplado |
| CRTBNDCPP | Criar programa C++ vinculado |
| CRTBNDRPG | Criar Programa de RPG Vinculado |
| CRTPGM | Criar programa |
| DLTPGM | Excluir programa |
| DSPPGM | Programa de exibição |
| DSPPGMREF | Exibir referências de programas |
| UPDPGM | Programa de Atualização |
| WRKPGM | Trabalhar com o programa |

**Comandos** CL  **usados com** **programas de serviço**

*Tabela 16. Comandos* CL  *usados com* *programas de serviço*

|  |  |
| --- | --- |
| **Comando** | **Nome** Descritivo |
| CHGSRVPGM | Programa de Serviço de Alteração |

© Direitos autorais IBM Corp. 1997, 2010

*Tabela 16. Comandos* CL  *usados com* *programas de serviço (continuação)*

|  |  |
| --- | --- |
| CRTSRVPGM | Criar programa de serviço |
| DLTSRVPGM | Excluir programa de serviço |
| DSPSRVPGM | Programa de serviço de exibição |
| RTVBNDSRC | Recuperar origem do fichário |
| UPDSRVPGM | Programa de Serviço de Atualização |
| WRKSRVPGM | Trabalhar com o Programa de Serviço |

**Comandos** CL  **usados com diretórios de vinculação**

*Tabela 17. Comandos* CL  *usados com diretórios de vinculação*

|  |  |
| --- | --- |
| **Comando** | **Nome** Descritivo |
| ADDBNDDIRE | Adicionar entrada de diretório de vinculação |
| CRTBNDDIR | Criar diretório de associação |
| DLTBNDDIR | Excluir diretório de vinculação |
| DSPBNDDIR | Exibir diretório de vinculação |
| RMVBNDDIRE | Remover entrada de diretório de vinculação |
| WRKBNDDIR | Trabalhar com o diretório de vinculação |
| WRKBNDDIRE | Trabalhar com entrada de diretório de vinculação |

**Comandos** CL  **usados com a linguagem de consulta estruturada**

*Tabela 18. Comandos* CL  *usados com a linguagem de consulta estruturada*

|  |  |
| --- | --- |
| **Comando** | **Nome** Descritivo |
| CRTSQLCI | Criar objeto ILE C da linguagem de consulta estruturada |
| CRTSQLCBLI | Criar objeto COBOL ILE da linguagem de consulta estruturada |
| CRTSQLCPPI | Criar objeto SQL ILE C++ |
| CRTSQLRPGI | Criar objeto de RPG ILE de linguagem de consulta estruturada |

|

**Comandos** CL  **usados com CICS**

*Tabela 19. Comandos* CL  *usados com CICS*

|  |  |
| --- | --- |
| **Comando** | **Nome** Descritivo |
| CRTCICSC | Criar objeto CICS® ILE C |
| CRTCICSCBL | Criar programa CICS COBOL |

**Comandos** CL  **usados com o depurador de origem**

*Tabela 20. Comandos* CL  *usados com o depurador de origem*

|  |  |
| --- | --- |
| **Comando** | **Nome** Descritivo |
| DSPMODSRC | Origem do módulo de exibição |
| ENDDBG | Encerrar depuração |

*Tabela 20. Comandos* CL  *usados com o depurador de origem (continuação)*

|  |  |
| --- | --- |
| STRDBG | Iniciar depuração |

**Comandos** CL  **usados para editar o arquivo de origem do idioma do fichário**

*Tabela 21. Comandos* CL  *usados para editar* a origem do idioma  *do fichário*

|  |  |
| --- | --- |
| **Comando** | **Nome** Descritivo |
| EDTF | Editar arquivo |
| STRPDM | Iniciar Gerente de Desenvolvimento de Programação |
| STRSEU | Utilitário Iniciar Entrada de Origem |
| **Nota:** Os seguintes comandos não executáveis podem ser inseridos no arquivo de origem do idioma do fichário: | |
| ENDPGMEXP | Fim da exportação do programa |
| EXPORTAÇÃO | Exportação |

Apêndice C. Comandos CL usados com objetos ILE

**186 Conceitos** de ILE IBM i 7.1

**Apêndice D.**  **Avisos**

Essas informações foram desenvolvidas para produtos e serviços oferecidos nos EUA.

A IBM pode não oferecer os produtos, serviços ou recursos discutidos neste documento em outros países. Consulte o representante IBM local para obter informações sobre os produtos e serviços atualmente disponíveis na sua área. Qualquer referência a um produto, programa ou serviço IBM não se destina a indicar ou implicar que apenas esse produto, programa ou serviço IBM pode ser usado. Qualquer produto, programa ou serviço funcionalmente equivalente que não infrinja qualquer direito de propriedade intelectual da IBM pode ser utilizado em seu lugar. No entanto, é responsabilidade do usuário avaliar e verificar a operação de qualquer produto, programa ou serviço que não seja da IBM.

A IBM pode ter patentes ou pedidos de patente pendentes que abranjam o assunto descrito neste documento. O fornecimento deste documento não lhe dá qualquer licença para essas patentes. Você pode enviar consultas de licença, por escrito, para:

Diretor de Licenciamento da IBM

IBM Corporação

North Castle Drive Armonk, Nova Iorque 10504-1785 Estados Unidos da América

Para consultas de licença sobre informações de byte duplo (DBCS), entre em contato com o Departamento de Propriedade Intelectual da IBM em seu país ou envie perguntas, por escrito, para:

Licenciamento de Propriedade Intelectual Direito Legal e de Propriedade Intelectual IBM Japão, Ltd.

3-2-12, Roppongi, Minato-ku, Tóquio 106-8711

**O parágrafo a seguir** **não se aplica ao Reino Unido ou** a  **qualquer outro país onde tais disposições sejam inconsistentes com** **a legislação local:** INTERNATIONAL BUSINESS MACHINES CORPORATION

FORNECE ESTA PUBLICAÇÃO "NO ESTADO EM QUE SE ENCONTRA" SEM GARANTIA DE QUALQUER TIPO, SEJA EXPRESSA

OU IMPLÍCITAS, INCLUINDO, MAS NÃO SE LIMITANDO A, AS GARANTIAS IMPLÍCITAS DE

NÃO VIOLAÇÃO, COMERCIALIZAÇÃO OU ADEQUAÇÃO A UM PROPÓSITO ESPECÍFICO. Alguns

os estados não permitem a isenção de responsabilidade de garantias expressas ou implícitas em determinadas transações, portanto, esta declaração pode não se aplicar a você.

Essas informações podem incluir imprecisões técnicas ou erros tipográficos. Alterações são feitas periodicamente nas informações aqui contidas; essas alterações serão incorporadas em novas edições da publicação. A IBM pode fazer melhorias e/ou alterações no(s) produto(s) e/ou no(s) programa(s) descrito(s) nesta publicação a qualquer momento, sem aviso prévio.

Quaisquer referências nestas informações a Web sites que não sejam da IBM são fornecidas apenas por conveniência e não servem, de forma alguma, como um endosso desses Web sites. Os materiais nesses Web sites não fazem parte dos materiais para este produto IBM e a utilização desses Web sites é por sua conta e risco.

A IBM pode usar ou distribuir qualquer uma das informações que você fornecer de qualquer maneira que considere apropriada sem incorrer em qualquer obrigação para com você.

Licenciados deste programa que desejam ter informações sobre ele com a finalidade de permitir: (i) a troca de informações entre programas criados de forma independente e outros programas (incluindo este) e (ii) a utilização mútua das informações que foram trocadas, devem contactar:

IBM Corporação

© Direitos autorais IBM Corp. 1997, 2010

Coordenador de Interoperabilidade de Software, Departamento YBWA

3605 Highway 52 N Rochester, MN 55901 Estados Unidos da América

Essas informações podem estar disponíveis, sujeitas a termos e condições apropriados, incluindo, em alguns casos, o pagamento de uma taxa.

O programa licenciado descrito neste documento e todo o material licenciado disponível para ele são fornecidos pela IBM sob os termos do Contrato de Cliente IBM, Contrato de Licença do Programa Internacional IBM , Contrato de Licença IBM para Código de Máquina ou qualquer contrato equivalente entre nós.

Quaisquer dados de desempenho aqui contidos foram determinados em um ambiente controlado. Portanto, os resultados obtidos em outros ambientes operacionais podem variar significativamente. Algumas medições podem ter sido feitas em sistemas de nível de desenvolvimento e não há garantia de que essas medições serão as mesmas em sistemas geralmente disponíveis. Além disso, algumas medidas podem ter sido estimadas por extrapolação. Os resultados reais podem variar. Os usuários deste documento devem verificar os dados aplicáveis para seu ambiente específico.

As informações relativas a produtos não IBM foram obtidas junto dos fornecedores desses produtos, dos seus anúncios publicados ou de outras fontes publicamente disponíveis. A IBM não testou esses produtos e não pode confirmar a precisão do desempenho, compatibilidade ou quaisquer outras reivindicações relacionadas a produtos não IBM. Perguntas sobre os recursos de produtos não-IBM devem ser dirigidas aos fornecedores desses produtos.

Todas as declarações relativas à direção ou intenção futura da IBM estão sujeitas a alterações ou retiradas sem aviso prévio e representam apenas metas e objetivos.

Essas informações contêm exemplos de dados e relatórios usados em operações comerciais diárias. Para ilustrá-los da forma mais completa possível, os exemplos incluem os nomes de indivíduos, empresas, marcas e produtos. Todos esses nomes são fictícios e qualquer semelhança com os nomes e endereços usados por uma empresa de negócios real é inteiramente coincidência.

LICENÇA DE DIREITOS AUTORAIS:

Essas informações contêm programas de aplicativo de exemplo na linguagem de origem, que ilustram técnicas de programação em várias plataformas operacionais. Você pode copiar, modificar e distribuir esses programas de amostra de qualquer forma sem pagamento à IBM, para fins de desenvolvimento, uso, marketing ou distribuição de aplicativos programas em conformidade com a interface de programação de aplicativos para a plataforma operacional para a qual os programas de exemplo são gravados. Estes exemplos não foram exaustivamente testados em todas as condições. A IBM, portanto, não pode garantir ou implicar confiabilidade, facilidade de manutenção ou função desses programas. Os programas de exemplo são fornecidos "COMO ESTÃO", sem garantia de qualquer tipo. A IBM não será responsável por quaisquer danos decorrentes da sua utilização dos programas de amostragem.

Cada cópia ou qualquer parte desses programas de amostra ou qualquer trabalho derivado, deve incluir um aviso de direitos autorais da seguinte forma:

© (nome da sua empresa) (ano). Partes deste código são derivadas da IBM Corp. Programas de Amostra. Copyright IBM Corp. \_enter the year or years\_. © Todos os direitos reservados.

Se você estiver visualizando essa cópia eletrônica de informações, as fotografias e ilustrações coloridas podem não aparecer.

**Informações sobre** a interface de programação

Esta publicação ILE Concepts documenta Interfaces de Programação pretendidas que permitem ao cliente escrever programas para obter os serviços do IBM i.

**188 Conceitos** ILE IBM i 7.1

**Marcas comerciais**

IBM, o logotipo IBM e ibm.com® são marcas comerciais ou marcas registradas da International Business Machines Corp., registradas em muitas jurisdições em todo o mundo. Outros nomes de produtos e serviços podem ser marcas comerciais da IBM ou de outras empresas. Uma lista atual de marcas comerciais da IBM está disponível na Web em [Direitos autorais e informações de marcas comerciais](http://www.ibm.com/legal/copytrade.shtml) em www.ibm.com/legal/copytrade.shtml.

Linux é uma marca registrada da Linus Torvalds nos Estados Unidos, em outros países ou em ambos.

Microsoft, Windows, Windows NT e o logotipo do Windows são marcas comerciais da Microsoft Corporation nos Estados Unidos, em outros países ou em ambos.

Java e todas as marcas comerciais baseadas em Java são marcas comerciais da Sun Microsystems, Inc. nos Estados Unidos, em outros países ou em ambos.

UNIX é uma marca registrada do The Open Group nos Estados Unidos e em outros países.

Outros nomes de empresas, produtos ou serviços podem ser marcas comerciais ou marcas de serviço de terceiros.

1. . Onde 4-4-16 = sizeof(int) – sizeof(long) – sizeof(pointer) [↑](#footnote-ref-1)
2. . Onde 4-4-8 = sizeof(int) – sizeof(long) – sizeof(pointer) [↑](#footnote-ref-2)
3. . A opção do compilador teraspace TERASPACE(\*YES \*TSIFC) está disponível nos compiladores ILE C e C++ para mapear automaticamente malloc(), free(), calloc() e realloc() para suas versões teraspace quando STGMDL(\*SNGLVL) é especificado. [↑](#footnote-ref-3)