Linker (mesclador de arquivos), Loader (carregador de arquivos)

Lucas

19 de setembro de 2015

142-147 Passos do Ligador:

(a) Coloca códigos e módulos simbolicamente na memória.

(b) Determina endereços de dados e labels de instruções.

(c) Procura as bibliotecas dos programas para achar as rotinas contidas nelas e que são usadas pelo programa em sí, combina ambas referências internas e externas. Resolve referências entre os arquivos (B3 Patterson).

Quando todas as referências estão resolvidas, o ligador então vai determinar as localizações na memória que cada módulo deverá ocupar.

B18-B19 Passos do Carregador:

(a) Lê o cabeçalho do executável, para determinar o tamanho do segmento de dados e texto.

(b) Cria um novo espaço de endereços para o programa.

(c) Copia instruções e dados para um novo espaço de endereços.

(d) Copia para a pilha os argumentos passados pera o programa.

(e) Inicializa registradores da máquina, em geral a maioria dos registradores são limpados.

(f) Pula para uma rotina start-up que copia os argumentos do programa da pilha para registradores e chama a rotina principal (`main`). Se ela terminar o programa termina com o `exit system call`.

2.31.1a As tabelas abaixo tem as informações para o ligador, mesclar os objetos na memória, resolvendo os endereços internos e externos.

2.31.1 [5] 2.12 Ligue os objetos → Procedure A com .text size de 0x140 e .data size de 0x40 → Procedure B com .text size de 0x300 e .data size de 0x50.[a]

Text Segment	Address	Instruction		Text Segment	Address	Instruction	
	400000	lbu $a0, 8000($gp)			400140	sw $a1, 8040($gp)	
	400004	jal 100050			400144	jal 100000	
Data Segment	10000000	(X)		Data Segment	10000040	(Y)	
	...	...			...	...	
Relocation Info	Address	Instruction Type	Dependency	Relocation Info	Address	Instruction Type	Dependency
	0	lbu (I type)	X		0	sw (I type)	Y
	4	jal (J type)	B		4	jal (J type)	A
Symbol Table	Address	Symbol		Symbol Table	Address	Symbol	
	—	X			—	Y	
	—	B			—	A	

2.31.1b Ligue os objetos → Procedure A com .text size de 0x140 e .data size de 0x40 → Procedure B com .text size de 0x300 e .data size de 0x50.

Text Segment	Address	Instruction		Text Segment	Address	Instruction	
	400000	lui $at, 0x1000			400140	sw $a0, 8050($gp)	
	400004	ori $a0, $at, 0x0000			400144	jmp 0x60	
	...	...			...	...	
	0x84	jr $ra			0x180	jal 0x100000	
	...	...			...	...	
Data Segment	10000000	(X)		Data Segment	10000050	(Y)	
	...	...			...	...	
Relocation Info	Address	Instruction Type	Dependency	Relocation Info	Address	Instruction Type	Dependency
	0	lui (I type)	X		0	sw (I type)	Y
	4	ori (I type)	X		4	jump (J type)	F00
					0x180	jal (J type)	A
Symbol Table	Address	Symbol		Symbol Table	Address	Symbol	
	—	X			—	Y	
					0x180	F00	
					—	A	

[a] Como você pode ver nunca haverá a possibilidade de um jal, j, concatenar alguma coisa além de PC+4[31-28] == 0000. Senão você estaria pulando para uma área de dados, no layout que estamos trabalhando aqui. É por isso que dá erro de acesso indevido a memória.