Painel ► Cursos ► INE5424-06208A (20221) ► Preliminaries ► E0: Compiling, linking, and loading review

Iniciado em	Tuesday, 19 Apr 2022, 18:27
Estado	Finalizada
Concluída em	Tuesday, 19 Apr 2022, 19:18
Tempo empregado	50 minutos 19 segundos
Notas	31,0/36,0

Avaliar 8,6 de um máximo de 10,0(86%)

1. Sobre a função Hello World:
a) Ao utilizar a assinatura no estilo "void main(void)", a função main não é 🗸 v considerada conformante com o padrão ANSI C
b) Um processo de usuário, e.g. a função "main()" 🗸 , precisa 🗸 aderir às definições POSIX em <errno.h>.</errno.h>
c) A função printf() é uma função da libc 🗸 🗸 .
d) A função main() possui v um stack frame, pois
é uma função como outra qualquer e terá um stack frame se nada for dito em contrário explicitamente durante a compilação 🔻 .
2. Sobre Modelo de memória:
a) Variáveis globais inicializadas não são zeradas pelo SO em tempo de carga, sendo o contrário 🗸 válido para variáveis globais não
inicializadas. Variáveis locais não inicializadas não são 🗸 zeradas pelo SO em tempo de carga. Variáveis locais estáticas (static)
são ✓ ✓ inicializadas como globais.
b) Variáveis complexas como estruturas e arrays,
são inicializadas como qualquer outra variável, pois C define um modelo rígido e fixo de memória 🗸 .
c) Memória dinâmica alocada com new() ou malloc() é alocada na heap 🗸 🗸 , a qual reside dentro do segmento de dados 🗸 do
processo. O operador new() de C++, com o alocador padrão, faz uso da função malloc() da libc, a qual interage com o SO através da system
call brk() ✓ ✓ para expandir o segmento de dados do processo e, com isto, sua heap ✓ ✓ .
2 Coher a formate From table and Linkship Format (FLD)
3. Sobre o formato Executable and Linkable Format (ELF):
a) Em relação a segmento e seções
b) As seções .text e .init não são alocadas no segmento de dados 🗸 .
c) As seções .data e .bss geralmente são alocadas no segmento de dados, sendo assumida como "0" e não figurando no objeto ELF
d) As constantes de um programa são coletadas em uma seção específica pelo compilador e carregadas pelo SO do arquivo ELF para o segmento de dados 🗸 🗶 .
e) O alinhamento em memória especificado em cada seção ELF não é 🗸 🗸 apenas uma otimização, sendo necessário
respeitá-las no SO.
f) A tabela de símbolos contida em um objeto ELF pode ser mantida 🗸 após as relocações durante a fase de ligação que produz
executáveis.
4. Em relação à carga de programas ELF e à criação de processos pelo SO:
a) A chamada de sistema fork() cria um novo processo, mas não carrega a imagem ELF do sistema de arquivos 🗸 🗸 , sendo
responsabilidade do execve() carregar a imagem 🗸 🗸 .
b) Não se pode afirmar que cada programa possui um endereço base de memória diferente 🗸 , fazendo com que
a MMU seja responsável de multiplexar a memória de forma segura 🗸 🗸 Anulada
c) A fração do segmento de dados correspondente à seção .bss é sempre zerada pelo SO no momento da carga do processo 🗸 .
d) A pilha de um processo é alocada dentro do segmento de dados, antes do .bss 💙 🗶 , não expresso 🗸 no ELF.
e) Um processo termina quando a função main() retorna ao escopo das funções de inicialização (CRTs) 🗸 🗸 , que então invocam a
chamada de sistema exit().
5. Considerando-se a compilação de programas C como o Hello World do tutorial:
a) O flag "-c" instrui o compilador a compilar um programa sem gerar 🗸 um executável.
b) O flag "-S" instrui o compilador a compilar o programa e emitir o código intermediário em formato assembly para o assembler da plataforma 🗸 🗸 .
c) O flag "-O" instrui o compilador otimizar o resultado da compilação, e não altera a semântica do programa 🗸 .
d) O flag "-g" instrui o compilador a instrumentar o objeto com informações de depuração, tendo o efeito colateral de produzir executáveis mais lentos

Questão 1
Parcialmente
correto
Atingiu 31,0 de
36,0

e) A ligação dos objetos ELF correspondentes a um programa escrito em C para produzir um executável ELF depende da libc por ser padrão da linguagem 🗸 .	
f) A libgcc é uma biblioteca interna do GCC com funções recorrentes otimizadas para a arquitetura alvo 🗸 .	
g) A ligação estática produz executáveis maiores do que a ligação dinâmica.	

■ Where is my Variable?

Seguir para...

Programming Language Performance Comparison by Gregory Alan Hildstrom ▶