Universidade Federal de Uberlândia (UFU)

Faculdade de Computação (FACOM)

Disciplina: GSI018 - Sistemas Operacionais

Prof. Rivalino Matias Jr.

Atividade Prática (Unidade II)

1. Utilizando dois compiladores diferentes de linguagem C, produza o código assembly para o programa abaixo. Analise as diferenças de cada código assembly produzido em comparação ao código C. A plataforma de SO é de livre escolha.

```
#include <stdio.h>
int i=3;
int j;
main(){
  int w;
  int z=3;
  printf("Hello World!\n");
  printf("%d%d%d%d",i,j,w,z);
}
```

2. A partir do código-fonte abaixo, gere o programa executável, execute-o e anote a senha que será exibida na tela. Observe que a rotina passcode() não faz parte do programa abaixo e nem da linguagem C. Sua implementação está disponível no arquivo objeto denominado "passcode.o" que acompanha esta lista. Para usar essa rotina é necessário incluir o arquivo de cabeçalho "passcode.h" que acompanha o "passcode.o". O arquivo "passcode.o" foi criado para funcionar apenas no sistema operacional Linux.

```
#include "passcode.h"
main(){
        char code[11];
        passcode(code);
        printf("%s",code);
}
```

3. O arquivo de programa "prog01.exe" (no diretório desta lista) possui três funções, main(), f1() e f2(), cujo código fonte em C está listado abaixo. Faça alterações **diretamente** no arquivo de programa (prog01.exe) para que ao ser executado a função main() chame primeiro f2() e depois f1(). O arquivo de programa prog01.exe foi criado para funcionar apenas no sistema operacional Linux.

```
#include <stdio.h>

f1(){ printf("F1");}
f2(){ printf("F2");}

main(){
        f1();
        f2();
}
```

4. A partir do código-fonte abaixo, crie os arquivos de programas "prog02.exe" e "prog03.exe". Execute cada programa comparando seus tempos de execução. Para isso, utilize o comando "time" (Linux) ou "PowerShell Measure-Command" (Windows). Abaixo exemplos de utilização.

```
/* prog02.c */
#include <stdio.h>
main(){
  int i;
  for(i=0;i<10000;i++)
     printf("A\n");
}

/* prog03.c */
#include <stdio.h>
#include<string.h>
main(){
  int i;
  char str[20001]="";
  for(i=0;i<10000;i+=2)
  {
       *(str+i)='A';
       *(str+i+1)='\n';
  }
  *(str+i)='\0';
  printf("%s",str);
}
```

```
C:\Users\johndoe>powershell -command "Measure-Command {prog02.exe}"
C:\Users\johndoe>powershell -command "Measure-Command {prog03.exe}"
```

Obs: Execute 11 vezes cada programa, descarte a primeira execução de cada programa, e tire a média dos demais 10 resultados de cada programa para ter um valor aproximado dos tempos de execução de cada um.

5. Fazer um programa, em linguagem C, para contar e imprimir o número total de arquivos armazenados em um disco rígido. Implementar e comparar o tempo de execução de três versões desse programa. A primeira versão deve ser programada como um único processo *singlthreaded*. A segunda versão deve ser programada como múltiplos processos *singlthreaded*, onde o número de processos (*n*) deve corresponder ao número de processadores do computador. Caso o computador tenha apenas um processador, então utilize $n = 2$. A terceira versão deve ser programada como múltiplos processos, tal como a segunda versão, contudo cada processo deve utilizar múltiplas *threads* (*mt*). O valor de *mt* deve ser 2. Na segunda e terceira versões, o algoritmo de busca e contagem de arquivos deve ser paralelizado; por exemplo, enquanto um processo conta os arquivos em uma parte do disco (ex. C:\ no Windows ou /dev/sda1 no Linux) o outro processo conta os arquivos em outra parte (ex. D:\ ou /dev/sda2). O mesmo aplica-se para múltiplas *threads*. A estratégia de paralelização do algoritmo de contagem de arquivos é de livre escolha, assim como a plataforma de SO escolhida para realizar esse exercício.

Pontuação (5,0 pontos): 1 ponto cada questão.