Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação em Sistemas Computacionais

Teste Global de 1ª Época, Inverno de 2013/2014

Nas questões em que não se indiquem explicitamente outras condições, considere as características do ambiente de referência da unidade curricular neste semestre.

1. [2] Escreva a função trim, que retorna uma versão da string str sem espaços no início e no fim. Não use outras funções da biblioteca *standard* para além de isspace, que deve utilizar para determinar se um carácter é considerado espaço. A *string* resultante residirá no mesmo espaço da *string* de entrada, que pode ser alterada. Minimize o número total de *bytes* escritos no espaço de destino.

```
int isspace(int c); /* from C library: returns non-zero if c is a white-space character, or zero otherwise */
char * trim(char * str);
```

2. [1,5] Das definições abaixo, seleccione as que podem corresponder à definição de variáveis locais e ordene-as de acordo com o espaço que cada uma ocupa em *stack* (para o mesmo tamanho pode usar qualquer ordem).

```
int newLine = '\n';                    short high_mark = 0x7F3D;
#define MAX 0x7FFFFFFF                 char * msg = "ISEL - LEIC - PSC";
double milli = 0.001;                  char STACK_SIZE = 24;
struct data { int x; int y; int z; };  unsigned char top = 255;
```

3. [2] Considere as definições abaixo. Apresente uma versão de choose_thinner em *assembly* IA-32.

```
typedef struct dimensions { int width; int height; int thickness; } Dimensions;
typedef struct device { unsigned id; Dimensions * dim; } Device;
Device * choose_thinner(Device * dv1, Device * dv2) {
   return (dv1->dim->thickness < dv2->dim->thickness) ? dv1 : dv2;
}
```

4. [3] Desenvolva, em *assembly* IA-32, a função filterObjs, que recebe em objs um *array* com n objectos de dim *bytes*, aplicando a função referida por eval aos endereços de cada um desses objectos. A função filterObjs preenche o *array* ptrs com ponteiros para os objectos de objs em que eval retorne um valor diferente de zero. Em cada chamada à função referida por eval, o ponteiro recebido em ctx é passado, inalterado, como segundo argumento. A função filterObjs retorna o número de ponteiros preenchidos em ptrs.

```
int filterObjs(void * objs, size_t n, size_t dim, void * ptrs[],
               void * ctx, int (*eval)(void * ptr, void * ctx));
```

5. [1] Os processadores Intel Core i7 usam uma cache de nível 2 por *core*, cada uma com organização 8-*way set associative* e capacidade de 256 KiB com linhas de 64 *bytes*. Apresentando os cálculos apropriados, determine qual é a menor distância possível entre os endereços de dois bytes armazenados em linhas distintas do mesmo *set* da cache indicada.

6. [1,5] Considere um programa, prog, cujo código fonte está organizado em 3 ficheiros fonte: prog.c, file1.c e file2.c. O programa utiliza uma biblioteca de ligação dinâmica, mylib.so, cujo código está presente nos ficheiros fonte mylibsrc1.c e mylibsrc2.c. Esta biblioteca é carregada implicitamente pelo sistema operativo, em conjunto com o executável base. Escreva um Makefile que permita construir, a partir do código-fonte, os artefactos binários indicados, com um mínimo de processamento.

7. [1] Observe o resultado do processamento de um ficheiro fonte, apresentado abaixo. Explique as mensagens de aviso e de erro e apresente problemas plausíveis que as possam originar.

```
isel@linuxvm:~/psc-1314-1/t1$ gcc -Wall -pedantic -std=c99 -g -o prog prog.c
prog.c: In function 'main':
prog.c:2: warning: implicit declaration of function 'strcmp'
prog.c:3: warning: implicit declaration of function 'func'
/tmp/cc7kRNcD.o: In function `main':
/home/isel/psc-1314-1/t1/prog.c:3: undefined reference to `func'
collect2: ld returned 1 exit status
```

8. [4] A função readInt lê um valor inteiro do *standard input*, deixando-o no endereço apontado por pval, retornando false se a leitura falhar por algum motivo. Uma instância do tipo ValCounts armazena em counters um *array* de len instâncias do tipo ValCounter. Cada instância de ValCounter armazena um valor inteiro (val) e uma contagem associada (count). A função countsForDistinctVals lê inteiros com readInt até que esta função retorne false e devolve uma instância de ValCounts com o número de ocorrências de cada valor distinto lido. A função freeValCounts liberta todo(s) o(s) espaço(s) de memória associado(s) a uma instância de ValCounts alocada dinamicamente. Implemente as funções countsForDistinctVals e freeValCounts. Se julgar necessário, pode criar funções auxiliares e o array de ValCounters pode consumir espaço extra.
Nota: se readInt ler 7 3 4 7 -1 7 3, o retorno pode ser { len = 4, counters = { { 7, 3 }, { 3, 2 }, { 4, 1 }, { -1, 1 } } }

```
bool readInt(int * pval) { return scanf("%d", pval) == 1; }
typedef struct val_counter { int val; size_t count; } ValCounter;
typedef struct val_counts { size_t len; ValCounter * counters; } ValCounts;
ValCounts * countsForDistinctVals();
void freeValCounts(ValCounts * vc);
```

9. [4] Considere o tipo abstracto IntVal, sem campos e com os métodos virtuais indicados em IntValOps. Considere ainda duas concretizações de IntVal: a primeira, IntVar, tem um campo interno de tipo int onde armazena o respectivo valor; a segunda, IntRef, contém um ponteiro para uma instância de qualquer tipo derivado de IntVal, para onde delega todas as operações. Apresente: as 3 estruturas em falta; a implementação de getVal para IntVar; a implementação de setVal para IntRef; uma implementação de printTo utilizável com qualquer IntVal; as tabelas de métodos de IntVar e de IntRef; e o código de IntRef_init, para inicializar uma instância de IntRef.
Nota: com um IntVar x de valor 3 e um IntRef y a referir x, após invocar setVal em y com 5 a chamada a printTo sobre x imprime 5

```
typedef struct intval IntVal;
typedef struct intvar IntVar;
typedef struct intref IntRef;

typedef struct intvalops IntValOps;

void IntVar_init(IntVar * iv, int iniVal);
void IntRef_init(IntRef * ir, IntVal * iniRef);
```

```
struct intvalops {
    int (*getVal)(IntVal * iv);
    void (*setVal)(IntVal * iv, int newVal);
    void (*printTo)(IntVal * iv, FILE * stream);
};
```

Duração: 2 horas e 30 minutos
ISEL, 16 de Janeiro de 2014