Universidade do Estado de Santa Catarina – Bacharelado em Ciência da Computação
Sistemas Operacionais – Prof. Rafael R. Obelheiro

Trabalho Prático — Apuração de Votos

1 Objetivo

O objetivo deste trabalho é projetar e desenvolver uma aplicação concorrente que realize a apuração de votos em uma eleição.

2 Descrição da Aplicação

A aplicação deverá realizar a apuração dos votos de uma eleição. Uma ou mais *threads* irão processar arquivos com votos, validar cada voto e colocar os votos válidos em uma fila para que sejam contabilizados por uma outra *thread*, que pode ser a própria função `main()`.

O programa recebe dois parâmetros na linha de comando, de acordo com o seguinte formato de invocação:

```
$ apuracao nthr nomearq
```

Os parâmetros são os seguintes:

- `nthr`: número de *threads* que processam arquivos de votos (≥ 1);
- `nomearq`: nome do arquivo contendo a lista de candidatos.

O nome de arquivo `nomearq` também é usado para compor o nome do arquivo com os votos de cada *thread*, no formato `nomearq-`*`idThread`*, onde *idThread* é um número inteiro de 1 a `nthr`, que representa o identificador da *thread*. Por exemplo, se o programa for invocado com

```
$ apuracao 4 abc
```

o arquivo `abc` conterá os candidatos válidos, e os arquivos `abc-1`, `abc-2`, `abc-3` e `abc-4` conterão os votos a serem processados pelas *threads* de 1 a 4.

O arquivo contendo a lista de candidatos válidos tem o formato mostrado abaixo (os nomes das colunas não estão contidos no arquivo):

N° do candidato	*Nome do candidato*
1234	Patrick Jane
2345	Teresa Lisbon
5678	Wayne Rigsby
1337	Grace van Pelt
2468	Kimball Cho

O número do candidato é um valor inteiro sem sinal (`unsigned int`), e o nome é uma *string* de até 30 caracteres. O número e o nome são separados por um número arbitrário de espaços em branco, e o nome registrado pelo programa deve começar no primeiro caracter diferente de espaço. O programa deve tratar a possibilidade de que os nomes contidos no arquivo tenham mais de 30 caracteres, truncando-os quando necessário.

Os arquivos de votos têm o formato mostrado abaixo:

N° do candidato
1234
4321
1234
1337

O número do candidato é um valor inteiro sem sinal (`unsigned int`). Cada linha representa um voto (portanto, um mesmo número pode ser repetido em várias linhas). Um voto é considerado válido quando refere-se a um candidato presente na lista de válidos.

O programa principal deve inicializar a lista de candidatos válidos a partir do arquivo e criar `nthr` *threads*. O processamento dos arquivos de voto só pode iniciar depois que todas as *threads* estiverem prontas. O programa deve então processar a lista de votos válidos, que será alimentada pelas *threads* à medida em que elas processam seus arquivos de votos. Por fim, o programa deve divulgar o resultado final da eleição, juntamente com algumas estatísticas. O pseudocódigo para o programa principal é o seguinte:

```
inicializa_lista_candidatos();
cria_threads();
enquanto (houver votos) {
   voto = desenfileira_voto();
   contabiliza_voto(voto);
}
espera_fim_threads();
mostra_estatisticas();
divulga_resultado();
```

Cada *thread* primeiramente inicializa suas variáveis e espera até que todas as demais *threads* estejam prontas. Na sequência, a *thread* deve processar o seu arquivo de votos, linha a linha. Caso o voto seja válido, ele deve ser inserido na lista para processamento pelo programa principal (todas as *threads* devem alimentar a mesma lista). Caso contrário, ele deve ser contabilizado como voto inválido. Após processar todos os seus votos, a *thread* encerra, retornando o número de votos processados e o número de votos inválidos para o programa principal. O pseudocódigo para uma *thread* é o seguinte:

```
total = 0;
invalidos = 0;
sinaliza_thread_pronta_e_espera_demais();
enquanto (houver votos a ler) {
   voto = le_voto(arquivo);
   total++;
   se (valido(voto))
      enfileira(voto);
   senao
      invalidos++;
}
retorna(total, invalidos)
```

Não há limites predefinidos para o número de candidatos válidos ou de votos em cada arquivo. Deve ser dada preferência ao uso de listas encadeadas ou outras estruturas alocadas dinamicamente. Soluções ineficientes usando vetores e `realloc()` serão penalizadas na nota.

As inserções e retiradas da lista de votos válidos devem ser concorrentes; não é aceitável, por exemplo, que a contabilização dos votos inicie apenas depois que todos eles tenham sido inseridos pelas *threads*. A solução implementada deve evitar condições de disputa entre as *threads*. A aplicação não deve estar sujeita a *deadlock* por conta do controle de concorrência. Soluções de espera ocupada não devem ser usadas.

Ao final de sua execução, as *threads* devem retornar ao programa principal o total de votos processados e o número de votos inválidos. Depois que todas as *threads* tiverem encerrado, o programa principal deverá exibir o total de votos processados, o total e o percentual de votos válidos, e o número máximo de nós observado na fila durante a execução (como as inserções e retiradas são concorrentes, esse número deve variar a cada execução). Na sequência, deve ser proferido o resultado da eleição, com os candidatos listados em ordem decrescente de votos, conforme o exemplo:

```
Total de votos processados: 204
Votos validos: 200 (98.04%)
Tamanho maximo da fila: 53 nos

Resultado final:
  Numero  Nome                        Votos validos    %
    1337  Grace van Pelt                         80  40.00
    1234  Patrick Jane                           52  26.00
    2345  Teresa Lisbon                          38  19.00
    5678  Wayne Rigsby                           25  12.50
    2468  Kimball Cho                             5   2.50
```

As *strings* de formato `fmtcabvoto` e `fmtvoto`, definidas no arquivo `utils.h`, podem ser usadas para produzir respectivamente o cabeçalho e cada linha da saída desejada com `printf()`.

Exemplos de arquivos de entrada e saída estão disponíveis no Moodle; seu programa deverá produzir a mesma saída (à exceção do tamanho máximo da fila, que varia a cada execução) para os arquivos de entrada fornecidos.

3 Apresentação e Avaliação

- O trabalho pode ser realizado individualmente ou em dupla.
- Um aspecto importante do *software* desenvolvido é o tratamento de erros, especialmente erros do usuário (como valores inválidos). Sua aplicação deve prever a possibilidade de erros, e tratá-los adequadamente. Por tratamento adequado, se entende que os erros devem ser detectados e informados ao usuário; a aplicação não pode ter sua execução abortada, mas não é necessário implementar estratégias sofisticadas de recuperação.
- O trabalho deve ser implementado em C, sob o sistema operacional Linux, usando as APIs para programação concorrente (Pthreads, memória compartilhada, semáforos POSIX) vistas em aula. Não é permitido usar outras bibliotecas que implementem porções significativas do trabalho. Em caso de dúvida, verifique **antes** com o professor.
- Deverá ser entregue **impresso** um relatório com até três páginas usando o formato de artigo da SBC (ponteiro disponível na página da disciplina). O relatório deve descrever de que forma foi implementado o controle de concorrência na aplicação, e qual a API utilizada. O código fonte dos programas, devidamente documentado, deve ser entregue **em formato ZIP** via Moodle.
  Obs.: é necessário respeitar apenas a formatação do modelo da SBC (tamanho de fonte, margens, etc.), não a estrutura do documento.
- O trabalho deverá ser entregue até **TERÇA-FEIRA, 25 DE SETEMBRO**. O Moodle aceitará submissões até 23h55min, sendo automaticamente bloqueado após a data limite. É **RESPONSABILIDADE DOS ALUNOS** garantir que o trabalho seja entregue no prazo.
- A critério do professor, alguns alunos/grupos poderão ser solicitados a realizar uma apresentação do trabalho. Essa apresentação incluirá uma demonstração do funcionamento do *software* desenvolvido e uma discussão do fonte.
- A nota do trabalho corresponde a 20% da média final. Alunos que não entregarem o trabalho, não cumprirem o prazo ou não participarem da apresentação (se houver) terão atribuída nota zero.
- Em caso de cópia de trabalhos, **todos** os alunos envolvidos terão atribuída nota zero. **É sua responsabilidade garantir que o seu trabalho não seja copiado indevidamente.**

Em caso de dúvidas ou dificuldades, entre em contato com o professor.