The Boys (versão 2024/2)

Este projeto foi inicialmente concebido pelos profs. Fabiano Silva, Luis Bona e Marcos Castilho para as disciplinas CI1001 e CI067, levemente inspirada na série de TV The Boys. Esta página descreve uma versão modificada do projeto original.

O problema

Este projeto consiste em simular um mundo com heróis, bases e missões. Cada herói possui um conjunto de habilidades (vôo, invisibilidade, visão de raios X, superaudição, etc.), um nível de experiência e outros atributos.

As bases estão espalhadas pelo mundo e os heróis se deslocam periodicamente entre elas. Cada base possui uma lotação máxima e uma fila de espera; ao chegar na base, o herói pode decidir entre esperar na fila ou desistir, viajando para outra base. Cada base possui um porteiro que gerencia a entrada/saída dos heróis e a fila de espera.

Esporadicamente surgem missões em locais aleatórios desse mundo; cada missão exige um conjunto de habilidades específicas. A missão pode ser atendida por uma equipe de heróis que se encontre em uma base, desde que os heróis da equipe tenham, em conjunto, as habilidades requeridas. Ao ser escolhido para participar de uma missão, cada herói recebe pontos de experiência.

Este projeto consiste em simular a dinâmica desse mundo durante um ano (525.600 minutos). Ao final, a simulação deve apresentar o número de missões cumpridas (pontos ganhos) por cada herói e outras estatísticas.

Simulação a eventos discretos

Simular é construir e executar um modelo computacional que imita de forma aproximada a realidade e sua evolução ao longo do tempo. A simulação é uma ferramenta muito usada no estudo de fenômenos naturais, como técnica de ensino (oferecendo para os alunos um ambiente onde é possível experimentar e errar sem as consequências do ambiente real) e também em muitos jogos eletrônicos. A simulação de eventos discretos (SED) é uma forma de simulação que modela a dinâmica de um sistema como uma sequência de eventos ocorrendo ao longo do tempo.

O modelo de simulação possui uma estrutura estática e uma estrutura dinâmica:

- **Estrutura estática**: descreve as entidades do sistema com seus atributos; o valor desses atributos em um dado momento define o **estado do sistema**.
- **Estrutura dinâmica**: descreve como o estado do sistema evolui; geralmente é definida através de uma **sequência de eventos** agendados ao longo do tempo, que alteram os atributos das entidades do sistema e podem criar novos eventos.

Um simulador é um programa que executa os eventos em sequência e evolui o modelo de simulação no tempo. Para isso, ele mantém um **relógio global**, representando o tempo atual no modelo, e uma **Lista de Eventos Futuros** (LEF), que é basicamente uma fila de prioridades ordenada por datas de ocorrência crescentes, como em uma agenda:

```
10:00 ir ao dentista
 : (aqui está no dentista)
11:45 voltar do dentista
12:00 almoçar
 : (aqui está almoçando)
13:00 ir à UFPR
13:30 ir à aula de Cálculo
 : (aqui está na aula)
15:20 sair da aula de Cálculo
15:30 ir à aula de Programação
 : (aqui está na aula)
17:30 sair da aula de Programação
18:00 voltar para casa
 : (aqui está voltando para casa)
...
```

O relógio do simulador geralmente é um número inteiro que representa o tempo no sistema simulado e não tem relação direta com o tempo físico. Dependendo do modelo, cada unidade de tempo simulado pode representar microssegundos (em uma simulação de circuitos lógicos), minutos (simulação de tráfego) ou mesmo milhares de anos (simulação geológica).

Executar a simulação consiste basicamente em processar os eventos em sequência. O processamento de um evento pode alterar o estado do sistema e criar novos eventos no futuro. O estado do sistema não é alterado entre dois eventos, por isso o relógio do simulador pode saltar diretamente de um evento para o próximo. Dessa forma, o ciclo básico de funcionamento do simulador é bem simples:

1. retirar o primeiro evento da lista de eventos
2. atualizar o relógio com o instante do evento
3. processar o evento:
    1. atualizar o estado do sistema
    2. agendar novos eventos porventura criados
4. repetir até concluir a simulação

Dessa forma, uma simulação pode ser traduzida no seguinte pseudocódigo:

```
iniciar as entidades e atributos do mundo
criar a fila de eventos futuros
criar os eventos iniciais
relógio = 0
repetir
    ev = 1º evento da lista de eventos futuros
    relógio = tempo (ev)
    caso tipo (ev) seja:
        EV1: executa evento 1
        EV2: executa evento 2
```

```
        EV3: executa evento 3
        ...
    fim
até o fim da simulação
apresentar resultados
```

Entidades

Nosso mundo simulado é composto de diversos tipos de entidades, cada uma com seus próprios atributos:

Herói (H): representa cada herói:

- ID: número inteiro ≥ 0 que identifica unicamente o herói;
- Habilidades: conjunto de habilidades que o herói possui. Cada habilidade é representada por um número inteiro ≥ 0;
- Paciência: número inteiro ≥ 0 que indica quão paciente uma pessoa é. Em nosso modelo, isso afeta as decisões de permanência em bases e filas;
- Velocidade: número inteiro ≥ 0 indicando a velocidade de deslocamento de um herói, que irá afetar o tempo de deslocamento entre as bases;
- Experiência: número inteiro ≥ 0 que indica o número de missões em que o herói já participou;
- Base: ID da base onde o herói se encontra no momento.

Base (B): Representa cada local frequentado pelos heróis para formar equipes; cada base tem uma localização em um plano cartesiano, tem uma lotação, pode estar cheia e possui uma fila de espera:

- ID: número inteiro ≥ 0 que identifica cada base;
- Lotação: número máximo de heróis naquela base;
- Presentes: conjunto dos IDs dos heróis que estão atualmente na base, constituem as equipes disponíveis para realizar missões;
- Espera: fila onde os heróis esperam para poder entrar na base;
- Local: localização da base (par de coordenadas inteiras X, Y ≥ 0).

Missão (M): representa cada missão:

- ID: número inteiro ≥ 0 que identifica a missão;
- Habilidades: conjunto de habilidades necessárias para cumprir a missão;
- Perigo: nível de perigo da missão; 

- Local: localização da missão (par de coordenadas inteiras X, Y ≥ 0).

Mundo (W): O mundo é definido pelas entidades acima e algumas informações gerais:

- NHerois: número total de heróis no mundo;
- Herois: vetor representando todos os heróis;
- NBases: número total de bases no mundo;
- Bases: vetor representando todas as bases;
- NMissoes: número total de missões a cumprir;
- Missoes: vetor representando todas as missões;
- NHabilidades: número de habilidades distintas possíveis;
- TamanhoMundo: coordenadas máximas do plano cartesiano (as coordenadas mínimas são 0, 0); vamos considerar que o mapa de nossa supercidade é representado por um plano cartesiano de tamanho tal que cada unidade representa **1 metro**;
- Relogio: número inteiro ≥ 0 indicando o tempo atual no mundo. Cada unidade de tempo no mundo simulado representa **1 minuto** de tempo real.

Eventos

Os eventos implementam as mudanças de estado que fazem evoluir a simulação. Cada evento tem um instante de ocorrência, pode consultar e alterar variáveis (atributos das entidades) e pode criar outros eventos no presente ou no futuro.

Evento CHEGA

Representa um herói H chegando em uma base B no instante T. Ao chegar, o herói analisa o tamanho da fila e decide se espera para entrar ou desiste:

```
CHEGA (T, H, B):

atualiza base de H

se há vagas em B e a fila de espera em B está vazia:
    espera = true
senão:
    espera = (paciência de H) > (10 * tamanho da fila em B)

se espera:
    cria e insere na LEF o evento ESPERA (agora, H, B)
senão:
    cria e insere na LEF o evento DESISTE (agora, H, B)
```

A Lista de Eventos Futuros (LEF) pode ser implementada usando o TAD “fila de prioridades” previamente implementado.

Evento ESPERA

O herói H entra na fila de espera da base B. Assim que H entrar na fila, o porteiro da base B deve ser avisado para verificar a fila:

```
ESPERA (T, H, B):

adiciona H ao fim da fila de espera de B
cria e insere na LEF o evento AVISA (agora, B)
```

A fila de espera de cada base pode ser implementada usando o TAD “lista de inteiros” previamente implementado.

Evento DESISTE

O herói H desiste de entrar na base B, escolhe uma base aleatória D e viaja para lá:

```
DESISTE (T, H, B):
```

```
escolhe uma base destino D aleatória
cria e insere na LEF o evento VIAJA (agora, H, D)
```

Evento AVISA

O porteiro da base B trata a fila de espera:

```
AVISA (T, B):

enquanto houver vaga em B e houver heróis esperando na fila:
    retira primeiro herói (H') da fila de B
    adiciona H' ao conjunto de heróis presentes em B
    cria e insere na LEF o evento ENTRA (agora, H', B)
```

Evento ENTRA

O herói H entra na base B. Ao entrar, o herói decide quanto tempo vai ficar e agenda sua saída da base:

```
ENTRA (T, H, B):

calcula TPB = tempo de permanência na base:
    TPB = 15 + paciência de H * aleatório [1...20]
cria e insere na LEF o evento SAI (agora + TPB, H, B)
```

Evento SAI

O herói H sai da base B. Ao sair, escolhe uma base de destino para viajar; o porteiro de B é avisado, pois uma vaga foi liberada:

```
SAI (T, H, B):

retira H do conjunto de heróis presentes em B
escolhe uma base destino D aleatória
cria e insere na LEF o evento VIAJA (agora, H, D)
cria e insere na LEF o evento AVISA (agora, B)
```

Evento VIAJA

O herói H se desloca para uma base D (que pode ser a mesma onde já está):

```
VIAJA (T, H, D):

calcula duração da viagem:
    distância = distância cartesiana entre a base atual de H e a base D
    duração = distância / velocidade de H
cria e insere na LEF o evento CHEGA (agora + duração, H, D)
```

Evento MORRE

O herói H morre no instante T. New!

- O herói é retirado da base B e libera uma vaga na base.
- O porteiro de B deve ser avisado da nova vaga.
- Eventos futuros de um herói morto devem ser ignorados.

```
MORRE (T, H, B):

retira H do conjunto de heróis presentes em B
muda o status de H para morto
cria e insere na LEF o evento AVISA (agora, B)
```

Evento MISSAO

Uma missão M é disparada no instante T. São características de uma missão:

- Cada missão ocorre em um local aleatório e requer um conjunto aleatório de habilidades; ambos são definidos durante a inicialização.
- Cada equipe é formada pelo conjunto de heróis presentes em uma base.
- Uma equipe está apta para a missão se a união das habilidades de seus heróis contém as habilidades requeridas pela missão.
- Deve ser escolhida para a missão a equipe da base mais próxima ao local da missão e que esteja apta para ela.
- Ao completar uma missão, os heróis da equipe escolhida ganham pontos de experiência.
- Um herói pode morrer ao participar de uma missão. New!
- Se uma missão não puder ser completada, ela é marcada como “impossível” e adiada de 24 horas.

Para simplificar o modelo, as missões são consideradas instantâneas e sem tempo de deslocamento. Por isso, neste modelo o herói nem sai da base. Um modelo mais completo deveria considerar as saídas para missões e os retornos, com suas respectivas durações.

```
MISSAO (T, M):

calcula a distância de cada base ao local da missão M
encontra BMP = base mais próxima da missão cujos heróis possam cumpri-la
se houver uma BMP:
    marca a missão M como cumprida
    para cada herói H presente na BMP:
        risco = perigo (M) / (paciência (H) + experiência (H) + 1.0)
        se risco > aleatório (0, 30):
            cria e insere na LEF o evento MORRE (agora, H)
        senão:
            incrementa a experiência de H
senão:
    cria e insere na LEF o evento MISSAO (T + 24*60, M) para o dia seguinte
```

Evento FIM

Encerra a simulação no instante T:

```
FIM (T):
  encerra a simulação
```

```
  apresenta estatísticas dos heróis
  apresenta estatísticas das bases
  apresenta estatísticas das missões
```

Estado inicial

Inicialização do mundo virtual:

```
Tempo inicial:         T_INICIO        = 0
Tempo final (minutos): T_FIM_DO_MUNDO  = 525600
Tamanho do mundo:      N_TAMANHO_MUNDO = 20000
Número de habilidades: N_HABILIDADES   = 10
Número de heróis:      N_HEROIS        = N_HABILIDADES * 5
Número de bases:       N_BASES         = N_HEROIS / 5
Número de missões:     N_MISSOES       = T_FIM_DO_MUNDO / 100
```

Inicialização de cada herói:

```
id          = número sequencial [0...N_HEROIS-1]
experiência = 0
paciência   = número aleatório [0...100]
velocidade  = número aleatório [50...5000]  // em metros/minuto = 3 Km/h a 300 Km/h
habilidades = conjunto com tamanho aleatório [1...3] de habilidades aleatórias
```

Observe que cada herói possui entre 1 e 3 habilidades distintas, escolhidas aleatoriamente entre as habilidades possíveis.

Inicialização de cada base:

```
id        = número sequencial [0...N_BASES-1]
local     = par de números aleatórios [0...N_TAMANHO_MUNDO-1]
lotação   = número aleatório [3...10]
presentes = conjunto vazio (com capacidade para armazenar IDs de heróis até a
lotação da base)
espera    = fila vazia
```

Inicialização de cada missão:

```
id          = número sequencial [0...N_MISSOES-1]
local       = par de números aleatórios [0...N_TAMANHO_MUNDO-1]
habilidades = conjunto com tamanho aleatório [6...10] de habilidades aleatórias
perigo      = número aleatório [0...100]
```

Eventos iniciais

Antes de iniciar a simulação, é necessário criar e inserir na LEF alguns eventos iniciais. Esses eventos serão em seguida processados e poderão gerar novos eventos, o que fará avançar a simulação.

Cada herói chegará em uma base aleatória em algum momento dos três primeiros dias de simulação:

```
para cada herói H:
    base  = número aleatório [0...N_BASES-1]
```

```
    tempo = número aleatório [0...4320]  // 4320 = 60*24*3 = 3 dias
    criar e inserir na LEF o evento CHEGA (tempo, H, base)
```

Cada missão deve ser agendada para ocorrer em algum momento da simulação:

```
para cada missão M:
    tempo = número aleatório [0...T_FIM_DO_MUNDO]
    criar e inserir na LEF o evento MISSÃO (tempo, M)
```

O evento FIM deve ser agendado para o instante final da simulação:

```
tempo = T_FIM_DO_MUNDO
criar e inserir na LEF o evento FIM (tempo)
```

Mensagens de Saída

Nenhuma divindade virtual se sente realizada sem conhecimento do que as pessoas de seu mundo estão fazendo. Por isso, nossa simulação deve imprimir algumas informações durante o processamento dos eventos. Essas mensagens também podem ser úteis no processo de depuração do código.

As seguintes mensagens devem ser geradas durante a execução dos eventos, com os formatos de impressão especificados:

Evento CHEGA

```
 45844: CHEGA  HEROI  0 BASE 2 ( 7/ 9) ESPERA
ou
 45844: CHEGA  HEROI  0 BASE 2 ( 7/ 9) DESISTE

%6d: CHEGA  HEROI %2d BASE %d (%2d/%2d) ESPERA
%6d: CHEGA  HEROI %2d BASE %d (%2d/%2d) DESISTE
```

Significado: no instante 45844 o herói 0 chega à entrada da base 2, que já tem 7 heróis presentes (sem contar ele) e lotação de 9 heróis. A mensagem informa também se o herói decidiu esperar para entrar ou desistiu.

Evento ESPERA

```
 45844: ESPERA HEROI  0 BASE 2 ( 4)

%6d: ESPERA HEROI %2d BASE %d (%2d)
```

Significado: no instante 45844 o herói 0 entra na fila de espera da base 2, que já tem 4 heróis aguardando na fila (sem contar ele).

Evento DESISTE

```
 45844: DESIST HEROI  0 BASE 2

%6d: DESIST HEROI %2d BASE %d
```

Significado: no instante 45844 o herói 0 desiste de entrar na base 2.

Evento AVISA

```
 45844: AVISA  PORTEIRO BASE 2 ( 7/ 9) FILA [ 30 17 04 23  0 ]
 45844: AVISA  PORTEIRO BASE 2 ADMITE 30
 45844: AVISA  PORTEIRO BASE 2 ADMITE 17

%6d: AVISA  PORTEIRO BASE %d (%2d/%2d) FILA [ %2d %2d ... ]
%6d: AVISA  PORTEIRO BASE %d ADMITE %2d
```

Significado:

- no instante 45844 o porteiro da base 2 é avisado; na base há 7 heróis, ela tem capacidade para 9 heróis e a fila de espera tem os heróis 30, 17, 04, 23 e 0, nessa ordem.
- no instante 45844 o porteiro da base 2 retira o herói 30 da fila de espera e o admite na base.
- no instante 45844 o porteiro da base 2 retira o herói 17 da fila de espera e o admite na base.

Evento ENTRA

```
45844: ENTRA  HEROI 30 BASE 2 ( 8/ 9) SAI 46101

%6d: ENTRA  HEROI %2d BASE %d (%2d/%2d) SAI %d
```

Significado: no instante 45844 o herói 30 entra na base 2, que passa a ter 8 presentes e lotação de 9; sua saída da base está agendada para o instante 46101.

Evento SAI

```
46101: SAI    HEROI 30 BASE 2 ( 7/9)

%6d: SAI    HEROI %2d BASE %d (%2d/%2d)
```

Significado: no instante 46101 o herói 30 sai da base 2, que passa a ter 7 heróis presentes e lotação de 9.

Evento VIAJA

```
46101: VIAJA  HEROI 30 BASE 2 BASE 6 DIST 6922 VEL 4763 CHEGA 46102

%6d: VIAJA  HEROI %2d BASE %d BASE %d DIST %d VEL %d CHEGA %d
```

Significado: no instante 46101 o herói 30 inicia uma viagem da base 2 à base 6, com distância 6922 m, velocidade 4763 m/min e chegada prevista no instante 46102.

Evento MORRE


```
46101: MORRE  HEROI 27 MISSAO 3186
```

```
%6d: MORRE  HEROI %2d MISSAO %d
```

Significado: no instante 46101 o herói 27 morre, após participar da missão 3186.

Evento MISSÃO

Cada evento MISSÃO deve gerar ao menos estas mensagens:

```
120037: MISSAO 4150 TENT 1 HAB REQ: [ 0 2 3 4 6 7 8 9 ]
... (msgs de depuração, vide abaixo)
120037: MISSAO 4150 CUMPRIDA BASE 7 HABS: [ 0 1 2 3 4 5 6 7 8 9 ]
ou, caso não exista base com equipe apta:
120037: MISSAO 4150 IMPOSSIVEL

%6d: MISSAO %d TENT %d HAB REQ: [ %d %d ... ]
%6d: MISSAO %d CUMPRIDA BASE %d HABS: [ %d %d ... ]
ou
%6d: MISSAO %d IMPOSSIVEL
```

Significado:

- Linha “MISSAO 4150 TENT 1 HAB REQ”: tentativa 1 de realizar a missão 4150, que requer as habilidades 0, 2, 3, 4, 6, 7, 8 e 9.
- Linha “MISSAO 4150 CUMPRIDA BASE 7”: a missão 4150 foi cumprida pela equipe da base 7, cujas habilidades reunidas são 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9.
- Linha “MISSAO 4150 IMPOSSIVEL”: não foi encontrada equipe para cumprir a missão 4150 nesta tentativa e ela foi reagendada.

Além das mensagens acima, **recomenda-se fortemente** gerar mensagens de depuração, detalhando como as bases são analisadas para encontrar a base mais próxima capaz de atender a missão:

```
120037: MISSAO 4150 BASE 1 DIST 6461 HEROIS [ 2 6 25 47 48 ]
120037: MISSAO 4150 HAB HEROI  2: [ 0 9 ]
120037: MISSAO 4150 HAB HEROI  6: [ 8 ]
120037: MISSAO 4150 HAB HEROI 25: [ 3 7 ]
120037: MISSAO 4150 HAB HEROI 47: [ 0 1 ]
120037: MISSAO 4150 HAB HEROI 48: [ 6 ]
120037: MISSAO 4150 UNIAO HAB BASE 1: [ 0 1 3 6 7 8 9 ]
...
120037: MISSAO 4150 BASE 7 DIST 9867 HEROIS [ 16 21 22 23 24 31 33 46 ]
120037: MISSAO 4150 HAB HEROI 16: [ 3 4 8 ]
...
120037: MISSAO 4150 HAB HEROI 46: [ 2 ]
120037: MISSAO 4150 UNIAO HAB BASE 7: [ 0 1 2 3 4 5 6 7 8 9 ]
120037: MISSAO 4150 CUMPRIDA BASE 7

%6d: MISSAO %d BASE %d DIST %d HEROIS [ %d %d ... ]
%6d: MISSAO %d HAB HEROI %2d: [ %d ... ]
%6d: MISSAO %d UNIAO HAB BASE %d: [ %d %d ... ]
```

Significado:

- Linha “BASE 1 DIST 6461”: a base 1 está distante 6461 metros da missão 4150 e tem os heróis 2, 6, 25, 47 e 48.
- Linha “HAB HEROI 2”: as habilidades do herói 2 são 0 e 9.

- ... (repete para os demais heróis presentes na base 1).
- Linha “UNIAO HAB BASE 1”: a união das habilidades dos heróis na base 1 é 0, 1, 3, 6, 7, 8 e 9.
- ... (repetir para as demais bases, até achar uma base apta).
- Observe que as bases são analisadas em ordem crescente de distâncias da missão.

Evento FIM

O evento FIM encerra a simulação, gerando um relatório com informações sobre heróis, bases e missões:

```
525600: FIM

HEROI  0 VIVO  PAC  32 VEL 3879 EXP  441 HABS [ 2 9 ]
... (repete para todos os heróis)

BASE  0 LOT  7 FILA MAX  3 MISSOES 434
... (repete para todas as bases)

EVENTOS TRATADOS: 285540
MISSOES CUMPRIDAS: 4925/5256 (93.7%)
TENTATIVAS/MISSAO: MIN 1, MAX 313, MEDIA 26.0
TAXA MORTALIDADE: 52.0%

HEROI %2d VIVO  PAC %3d VEL %4d EXP %4d HABS [ %d %d ... ]
HEROI %2d MORTO PAC %3d VEL %4d EXP %4d HABS [ %d %d ... ]
BASE %2d LOT %2d FILA MAX %2d MISSOES %d
EVENTOS TRATADOS: %d
MISSOES CUMPRIDAS: %d/%d (%.1f%%)
TENTATIVAS/MISSAO: MIN %d, MAX %d, MEDIA %.1f
TAXA MORTALIDADE: %.1f%%
```

Significado:

- O herói 0 está vivo, tem paciência 32, velocidade 3.879 Km/h, experiência 441 e possui as habilidades 2 e 9.
- A base 0 tem capacidade/lotação para 7 heróis, sua fila de espera teve no máximo 3 heróis e ela participou de 434 missões.
- Nesta simulação foram processados 285.540 eventos.
- Foram cumpridas 4.925 das 5.256 missões geradas (93,7% de sucesso).
- As missões tiveram entre 1 e 313 tentativas, a média foi de 26 tentativas/missão.
- 52% dos heróis morreram nas missões.

Arquivos iniciais

O arquivo theboys.tgz contém os arquivos necessários para iniciar o desenvolvimento do projeto:

- `conjunto.h`: interface do TAD Conjunto de Inteiros (**não deve ser alterado**)
- `conjunto.o`: implementação do TAD Conjunto de Inteiros (arquivo-objeto)
- `fprio.h`: interface do TAD Fila de Prioridades (TP5) (**não deve ser alterado**)
- `fprio.c`: implementação do TAD Fila de Prioridades (TP5)
- `lista.h`: interface do TAD Lista de Inteiros (TP4), (**não deve ser alterado**)
- `lista.c`: implementação do TAD Lista de Inteiros (TP4)
- `makefile`: arquivo do Make para a compilação
- `theboys.c`: esqueleto do programa principal, a completar

Requisitos

São requisitos para atribuição de notas a este trabalho:

- Uso de um arquivo `makefile` para facilitar a compilação.
- Os professores executarão `make` e deverão obter o arquivo executável funcional com a sua solução.
- O executável, cujo nome deverá ser `theboys`, deverá estar no subdiretório `theboys/`;
- Ao compilar, incluir as opções `-Wall -Werror -Wextra -std=c99 -g`.
- Se não compilar, o trabalho vale zero.
- Haverá desconto por cada *warning*.

Sobre o arquivo de entrega:

- Deve estar no formato TAR comprimido (`theboys.tgz`);
- O arquivo deve ser criado considerando-se que existe um diretório com o nome do trabalho. Por exemplo, este trabalho se chama *The Boys*, então seu arquivo `.tgz` deve ser criado assim:
    - Estando no diretório `theboys/`, execute:

```
make clean
cd ..
tar zcvf theboys.tgz theboys
```

- Desta maneira, quando os professores abrirem o arquivo compactado, terão garantidamente o diretório correto da entrega para poder fazer a correção semiautomática.
- O que colocar no arquivo `theboys.tgz`?
    - Todos os arquivos que são necessários para a compilação; se você usa outros arquivos além dos especificados, coloque-os também.
    - Minimamente, ele deve conter todos os arquivos `.c`, `.h` e o `makefile`;
    - Não envie programas de teste, arquivos de saída e outros arquivos desnecessários para a correção.
- Os professores testarão seu programa em uma máquina do Departamento de Informática (por exemplo, `cpu1`). Por isso, antes de entregar seu trabalho teste-o em máquinas do DINF para garantir que ele funcione bem.

Dicas

Organize seu desenvolvimento da seguinte forma:

- Desenvolva os TADs:
    1. Implemente e teste um módulo de TAD por vez
    2. Em cada módulo, implemente e teste uma função por vez
    3. Implemente as funções nesta ordem: `_imprime`, `_cria`, `_insere`, `_retira`, ...
    4. Em cada função, escreva e teste um bloco por vez
    5. Comece sempre pela parte mais simples
    6. Use os programas de teste e o Valgrind para se assegurar que o módulo está correto, antes de iniciar o próximo módulo.

- Desenvolva o programa `theboys.c`:
    1. Defina as estruturas de dados necessárias
    2. Implemente a criação e destruição dessas estruturas de dados
    3. Implemente a criação dos eventos iniciais
    4. Defina uma função para cada evento, contendo inicialmente apenas um `printf` do mesmo
    5. Implemente o laço principal de simulação, que chama as funções dos eventos
    6. Implemente e teste uma função de evento por vez (deixe a missão por último)
    7. Analise a saída para ver se a sequência de eventos parece correta

8. Use o Valgrind para verificar eventuais problemas de memória

Pode ser complexo depurar erros de lógica no projeto usando suas dimensões completas; por isso, sugere-se trabalhar com **números pequenos** de heróis, bases, missões e tempo, para gerar menos eventos, até conseguir entender e resolver os erros.

O arquivo `makefile`, além dos alvos usuais, contém o alvo `make tgz`, que constrói o arquivo final para entregar o projeto.

Uma simulação pode processar um grande número de eventos e gerar uma saída enorme.

Este arquivo
contém um exemplo simplificado de saída, com as primeiras 5.000 linhas e as últimas 5.000 linhas de uma execução cuja saída completa tem 6,3 milhões de linhas.

Devido ao seu tamanho, a análise manual de uma saída de simulação completa pode ser inviável. Por isso, aconselha-se o uso de ferramentas do *shell* UNIX para filtrar as linhas de interesse na saída. O comando `grep` permite filtrar linhas que contenham uma determinada sequência de caracteres (*string*).

Por exemplo, para filtrar as linhas da saída que mostrem eventos relacionados ao herói 37, pode-se usar este comando:

```
$ ./theboys | grep "HEROI 37" | more
  1303: CHEGA  HEROI 37 BASE 7 (10/16) ESPERA
  1303: ESPERA HEROI 37 BASE 7 (10)
  1303: ENTRA  HEROI 37 BASE 7 (11/16) SAI 1928
  1928: SAI    HEROI 37 BASE 7 ( 8/16)
  1928: VIAJA  HEROI 37 BASE 7 BASE 1 DIST 6536 VEL 3082 CHEGA 1930
  1930: CHEGA  HEROI 37 BASE 1 ( 6/25) ESPERA
...
```

Para filtrar as linhas relativas ao evento CHEGA na base 3:

```
$ ./theboys | grep ": CHEGA" | grep "BASE 3" | more
     8: CHEGA  HEROI 36 BASE 3 ( 0/29) ESPERA
   174: CHEGA  HEROI 34 BASE 3 ( 1/29) ESPERA
   275: CHEGA  HEROI  9 BASE 3 ( 1/29) ESPERA
   529: CHEGA  HEROI 34 BASE 3 ( 1/29) ESPERA
   593: CHEGA  HEROI 34 BASE 3 ( 1/29) ESPERA
   597: CHEGA  HEROI 44 BASE 3 ( 2/29) ESPERA
...
```

Resultados esperados

Por ser uma simulação e usar muitos números aleatórios, os resultados finais podem variam muito entre as implementações. A tabela abaixo traz valores estatísticos obtidos de 5.000 execuções do programa, com várias sementes aleatórias. Ela fornece uma ideia dos resultados esperados:

Resultado	Média	Mínimo	Máximo	Coef variação
Eventos tratados	382.943	178.980	1.123.966	37,5%
Taxa de mortalidade	40%	18%	64%	17,6%
Missões cumpridas	85,9%	5,3%	99,5%	17,0%

Resultado	Média	Mínimo	Máximo	Coef variação
Média de tentativas por missão	37,7	4,0	171,6	74,1%