Visão geral da tarefa

Para este trabalho, você tem a tarefa de descobrir uma senha secreta. Uma senha aleatória seria impossível de adivinhar. Felizmente, você tem algumas pistas para descobrir a senha:

- Você sabe que a senha possui 12 caracteres de comprimento.
- A senha pode conter qualquer caractere na faixa de 33 a 126 em ASCII.
- Você tem uma função chamada `checkPassword` (anexada a esta tarefa) que pode ser usada para descobrir a senha. Infelizmente, ela só pode verificar 3 caracteres consecutivos por vez. Você fornece um palpite de três caracteres (const char *password) e a posição na senha que deseja verificar (int start). A função retornará `0` se o palpite estiver correto, `-1` se não estiver correto.

Para descobrir a senha, você terá que usar força bruta para verificar todas as combinações possíveis de 12 caracteres. Você fará isso verificando três caracteres por vez: caracteres da posição 0-2, posição 3-5, posição 6-8 e posição 9-11. Quando seu programa encontrar uma correspondência, ele deve imprimi-la na tela.

Para tornar a tarefa de descoberta mais rápida, você dividirá a tarefa entre 4 processos e demonstrará que a bifurcação realiza a tarefa com mais rapidez.

Objetivo

Os objetivos deste trabalho são os seguintes:

a) Aprender sobre criação e controle de processos em um ambiente Linux.

b) Obter experiência com as chamadas de sistema `fork()`, `getpid()`, `getppid()` e `wait()`.

Instruções

Anexado a esta tarefa estão dois arquivos. Nenhum desses arquivos deve ser modificado:

- `checkPassword.h` - Um arquivo de cabeçalho contendo o protótipo para `checkPassword()`
- `checkPassword.o` - O arquivo de objeto para `checkPassword()`

Você escreverá um programa chamado `tarefa1.c`. Este programa irá:

- Aceitar um argumento opcional de linha de comando `-f` para indicar se o fork() deve ser executado ou não.
- Se a bifurcação não for solicitada, ele verificará os caracteres 0-2, 3-5, 6-8 e 9-11 nessa ordem.
- Se a bifurcação for solicitada
    - O processo pai gerará quatro processos filhos e aguardará a conclusão de todos os quatro processos filhos antes de sair.
        - Filho 1 verificará os caracteres 0-2.
        - Filho 2 verificará os caracteres 3-5.
        - Filho 3 verificará os caracteres 6-8.

        - Filho 4 verificará os caracteres 9-11.
- Seu programa deve imprimir os PIDs de todos os processos pai e filho.
- Quando três caracteres da senha forem encontrados, basta imprimi-los na tela. No caso de bifurcação, isso significa que a senha provavelmente será impressa fora de ordem. Isso é normal porque alguns processos encontrarão sua parte da senha mais rápido que outros. O objetivo é a velocidade. A ordem você pode descobrir manualmente.

Saída

Para demonstrar o tempo de processamento, use o comando `time`.

Por exemplo, supondo que a senha seja `abcdef123456`, executando:

`time ./tarefa1` deveria produzir a seguinte saída:

```
Processo 668970 com pai 666832 quebrando a senha...
Processo 668970 com pai 666832 encontrou 0-2: abc
Processo 668970 com pai 666832 encontrou 3-5: def
Processo 668970 com pai 666832 encontrou 6-8: 123
Processo 668970 com pai 666832 encontrou 9-11: 456

real    1m50.246s
user    0m3.109s
sys     0m14.690s
```

Executando `time ./tarefa1 -f` deve produzir a seguinte saída:

`Processo` 668954 com pai 666832 `quebrando a senha`...
`Processo` 668956 com pai 668954 encontrou 3-5: def
`Processo` 668955 com pai 668954 encontrou 0-2: abc
`Processo` 668957 com pai 668954 encontrou 6-8: 123
`Processo` 668958 com pai 668954 encontrou 9-11: 456

real 0m43.624s
user 0m2 .391s
sys 0m13.218s

(Nos exemplos mostrados, 668... é o PID obtido usando getpid() e getppid(). Esses números, é claro, serão diferentes para você):

Dicas úteis:

- Se você não se lembra de como ler argumentos de linha de comando em C, consulte:
    - https://www.tutorialspoint.com/cprogramming/c_command_line_arguments.htm

Para usar a função checkPassword:

1. Certifique-se de incluir `"checkPassword.h"` em seu arquivo fonte.
2. Certifique-se de vincular o arquivo objeto `checkPassword.o` ao compilar:

gcc -o tarefa1 checkPassword.o tarefa1.c

Chame a função `checkPassword` com uma string nula terminada de 4 caracteres e a posição em que deseja iniciar a pesquisa. Por exemplo:

```
char input[4] = "abc"; // String nula terminada: abc\0
// ...
result1 = checkPassword(input, 0); // Testa a senha abc???????????2222?
// ...
result2 = checkPassword(input, 3); // Testa a senha ???abc?????2????
```

Enviar arquivo Makefile para compilar seu programa.