Universidade Presbiteriana Mackenzie  
Ciência da Computação  
Sistemas Operacionais – Turma 04P11  
Alan Meniuk Gleizer – 10416804  
Caio Vinicius Corsini Filho – 10342005

Relatório Projeto 01 – Uso do Mutex

Para funcionamento correto do programa proposto, foi necessário chamar a função pthread\_mutex\_lock dentro de toda função que modifica o valor da variável balance. Isso garante que apenas uma thread pode acessar e modificar o valor da variável por vez. O bloqueio do mutex foi seguido sempre pela função pthread\_mutex\_unlock, para liberar alterações por outras threads.

Mais especificamente, esse ajuste foi realizado em dois pontos do código:

void deposit(BankAccount \*account, int amount) {  
 pthread\_mutex\_lock(&account->mutex); // lock o mutex antes de modificar a variável balance}  
 account->balance += amount;  
 printf("Depositado: %d, Saldo Atual: %d\n", amount, account→balance);  
 pthread\_mutex\_unlock(&account->mutex); // destranca o mutex após o fim da função  
}

void withdraw(BankAccount \*account, int amount) {  
 pthread\_mutex\_lock(&account->mutex); // lock o mutex antes de modificar a variável balance  
 if (account->balance >= amount) {  
 account->balance -= amount;  
 printf("Sacado: %d, Saldo Atual: %d\n", amount, account→balance);  
 } else {  
 printf("Saldo Insuficiente para saque: %d\n", amount);  
 }  
 pthread\_mutex\_unlock(&account->mutex); // destranca o mutex após o fim da função  
}

Abaixo foram incluídas capturas de tela de diversas execuções do código original e do código ajustado, nas quais é possível observar a eficácia do ajuste realizado. No código original, o valor final da balance é inconsistente devido ao acesso simultâneo do saldo de conta por múltiplas threads. Na versão ajustada, o valor final é sempre 1500.

Capturas de tela da execução  
Código original (sem uso correto do mutex)

Teste 01

  
Teste 02

  
Teste 03



Código ajustado (com uso correto do mutex)

Teste 01



Teste 02



Teste 03

