A) Sim, ocorre mudança na ordem das threads. E isso ocorre por conta do escalonamento, mas o que é escalonamento? Escalonamento de processos é o ato de realizar o chaveamento dos processos ativos, de acordo com regras bem estabelecidas, de forma que todos os processos tenham chance de utilizar a UCP. O escalonador é a parte do SO encarregada de decidir entre os processos prontos, qual será colocado em execução.

   Desse modo, ele cria vários fluxos de execução e cada execução tem uma ordem de execução aleatória que depende totalmente do sistema operacional e como ele escalona as threads. Portanto, não há como antever qual o resultado, de fato, será exibido por conta da

B) pthread é uma função que cria um novo fluxo, uma thread que vai executar a função PrintHello e o identificador da thread vai ser retornado na variável tid_sistema[thread] e então a thread vai ser criada.
   A partir do momento que a função pthread termina, não temos mais controle sobre em que momento exatamente a thread vai executar. A sua função é colocar a thread na fila de tarefas para o SO executar e assim ele vai usar a sua política de escalonamento.
   A diferença entre o primeiro programa e nesse, é que este programa possui identificação. E, dessa forma, conseguimos perceber que a maneira como é escalonada a função PrintHello é aleatória e imprevisível a fim de conseguir o seu processamento e imprimir a sua mensagem .
   Portanto, se há mais de um processador na máquina, nós não temos, dentro do programa, como controlar as ações do dispositivo, se continua com a execução da main e depois escalona a thread ou se interrompe a main e escalona a thread para executar na frente,logo saiu do escopo da minha aplicação, a biblioteca não tem funcionalidade para permitir esse tipo de gerência para o usuário.

C) Sim, o objetivo do programa era a função receber mais de um argumento, sendo indiferente seu tipo, logo o programa tem êxito no objetivo.O código utiliza o método de estrutura de dados - “ typedef Struct ” - onde dentro da estrutura é colocado todos os campos que eu preciso a fim de conseguir o resultado, na main é preenchida as variáveis da estrutura e depois é passado o endereço da variável que é a estrutura para a função, a diferenciação é automática no caso de estruturas. Desse modo, temos a possibilidade de fazer listagem de threads em sua ordem crescente até chegar na sua última.
D) Sim, a mudança que ocorre se configura na diferente ordenação da execução das threads (aleatória), no entanto a impressão da thread principal final é sempre a última a ser executada. Isso acontece devido a inserção da função pthread_join() que possui uma característica de bloquear o fluxo de execução

de threads por meio do identificador, na assinatura da função, assim que ele terminar sua execução é liberado o fluxo. O pthread_join espera todas as threads chegarem ao final da execução da função que elas receberam lá no pthread_create, onde elas sinalizam por meio do pthread_exit() que sua execução chegou ao fim, com o auxílio do loop a função consegue esperar todas as threads terminarem e, desse modo, a função seguinte dela só será executada depois que todas as threads terminarem, com o objetivo de quando quisermos pegar o resultado final do processamento de todas as threads após elas terminarem para determinados fins.