1. Vamos analisar o código e prever as ocorrências dos caracteres A, B, C, D e E:

O programa cria um processo filho usando a chamada de sistema `fork()`. O processo pai e o processo filho compartilham o mesmo código, e a execução é concorrente. Aqui está a previsão e o resultado das ocorrências:

```plaintext
Ocorrências previstas:

A: 2 vezes (uma do processo pai e uma do processo filho)

B: 1 vez (somente no processo filho)

C: 2 vezes (uma do processo pai e uma do processo filho)

D: 1 vez (somente no processo pai)

E: 2 vezes (uma do processo pai e uma do processo filho)
```

O resultado observado pode variar, dependendo da programação do sistema operacional. No entanto, geralmente, a saída é semelhante à previsão. O resultado observado deve ser algo parecido com:

```plaintext
A

C

B

D

E
```

2. Aqui está um programa em C que cria N processos filhos e imprime seus PIDs e PPIDs:

```c
#include <stdio.h>

#include <unistd.h>
```

```
int main() {
    int N;
    printf("Digite o número de processos a serem criados: ");
    scanf("%d", &N);

    for (int i = 0; i < N; i++) {
        pid_t pid = fork();

        if (pid == 0) {
            printf("Filho - PID: %d, PPID: %d\n", getpid(), getppid());
            break; // Para evitar que os filhos criem mais processos
        }
    }

    if (getpid() != 1) {
        // Aguarda todos os filhos terminarem
        for (int i = 0; i < N; i++) {
            wait(NULL);
        }
    }

    return 0;
}
```

Este programa solicita ao usuário o número de processos a serem criados, cria os processos, imprime seus PIDs e PPIDs e aguarda todos os filhos terminarem antes de encerrar.

3. Aqui está um programa em C que cria uma cadeia de processos onde cada processo cria o próximo na cadeia. Os PIDs e PPIDs são impressos na ordem inversa da criação:

```c
#include <stdio.h>
#include <unistd.h>

int main() {
   int N;
   printf("Digite o número de processos na cadeia: ");
   scanf("%d", &N);

   for (int i = 1; i <= N; i++) {
      pid_t pid = fork();

      if (pid == 0) {
         printf("Processo %d - PID: %d, PPID: %d\n", i, getpid(), getppid());
      } else {
         // O processo pai espera o filho terminar para criar o próximo
         wait(NULL);
         break; // Evita que o pai crie mais processos
      }
   }

   return 0;
}
```

Este programa permite ao usuário definir o número de processos na cadeia, cria os processos em sequência e imprime seus PIDs e PPIDs na ordem inversa da criação.