Betriebssysteme

Übung 3 ProcessControl

Gruppenmitglieder: Johannes Theiner
Semester: Sommersemester 2019
letzte Änderung: 04.04.2019

1 Level A

1. New → Ready freie Rechenkapazität
2. New → Ready/Suspend Keine freie Rechenkapazität
3. Ready → Running erster in Queue
4. Ready → Ready/Suspend andere Prozesse haben Vorrang
5. Ready/Suspend → Ready Rechenkapazität frei
6. Blocked → Ready externes Event ausgelöst (Tastatur etc.)
7. Blocked → Blocked/Suspend weitere Prozesse in Warteschlange
8. Blocked/Suspend → Ready/Suspend externes Event ausgelöst
9. Blocked/Suspend → Blocked Rechenkapazität frei
10. Running → Ready Rechenzeit abgelaufen
11. Running → Ready/Suspend Prozess unterbrochen
12. Running → Blocked warten auf Resource
13. Any State → Exit Prozess terminiert
14. New → Blocked Prozess weiß nicht ob Hardware benötigt wird
15. New → Blocked/Suspend Prozess weiß nicht ob Hardware benötigt wird
16. New → Running Prozess würde sich in der Queue vordrängeln
17. Ready → Blocked Prozess weiß nicht ob Hardware benötigt wird
18. Ready → Blocked/Suspend Prozess weiß nicht ob Hardware benötigt wird
19. Ready/Suspend → Blocked Anfrage an Hardware kann nicht gestellt werden
20. Ready/Suspend → Blocked/Suspend Anfrage an Hardware kann nicht gestellt werden

21. Ready/Suspend → Running Prozess würde sich vordrängeln
22. Blocked → Ready/Suspend Es müssten zwei Event gleichzeitig auftreten
23. Blocked → Running Prozess würde sich vordrängeln
24. Blocked/Suspend → Ready zwei Events gleichzeitig
25. Blocked/Suspend → Running zwei Events gleichzeitig
26. Running → Blocked/Suspend zwei Events gleichzeitig
27. Exit → Any State Prozess kann auf keine Resourcen mehr zugreifen da diese bereits aufgeräumt wurden

2 Level B

```
#include <iostream>
#include <zconf.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

void runProcess(char *process);

int main(int argc, char *argv[]) {
    if (argc != 2) {
        std::cout << "falsche Anzahl von Argumenten" << std::endl;
        return 1;
    }
    int count = std::stoi(argv[1]);
    char *toBeInvoked = getenv("TOBEINVOKED");
    if (toBeInvoked == nullptr) {
        std::cout << "nichts in TOBEINVOKED" << std::endl;
        return 1;
    }

    for (int i = 0; i < count; ++i) {
        runProcess(toBeInvoked);
    }
```

```
    return 0;
}

void runProcess(char * process) {
    pid_t pid = fork();
    if(pid == 0) {
        execlp(process, process, nullptr, nullptr);
        exit(1);
    }else {
        waitpid(pid, nullptr, 0);
    }
}
```

3 Level C

```
#include <iostream>
#include <fstream>
#include <zconf.h>
#include <sys/wait.h>
#include <sys/types.h>

int main(int argc, char *argv[]) {
    if (argc != 2) {
        std::cout << "falsche Anzahl von Argumenten" << std::endl;
        return 1;
    }
    int removeStatus = remove("test.txt");
    if(removeStatus) {
        std::cout << "Fehler beim entfernen der Datei :" << removeStatus << std::endl;
    }

    int count = std::stoi(argv[1]);
    for (int i = 0; i < count; ++i) {
        std::ofstream stream;
        stream.open("test.txt", std::ofstream::app);
        stream << "Hello World" << std::endl;
        stream.close();
    }
    std::cout << "Datei geschrieben" << std::endl;

    return 0;
```

```
}
```