

**Usman Institute of Technology**

**Department of Computer Science Fall 2022**

Name: Muhammad Waleed

Roll no: 20B-115-SE

Course: Operating Systems (CS312)

Course Instructor: Ma’am Shabina Mushtaq

Date: 24-Nov-2022

# FCFS (with arrival time 0):

import os

try:

    from rich.console import Console

    from rich.table import Table

except ModuleNotFoundError:

    os.system("pip install rich")

    from rich.console import Console

    from rich.table import Table

console = Console()

table = Table(show\_header=True, header\_style="bold magenta")

os.system("cls")

nprocess = int(input("Enter number of processes: "))

processes = []

CT = []

TAT = []

WT = []

for i in range(nprocess):

    b = int(input("Burst Time: "))

    processes.append(["P"+str(i+1), 0, b])

# sort According to arrival time

processes.sort(key=lambda x: x[1])

# Calculting Completion time

for i in range(len(processes)):

    if i == 0:

        CT.append(processes[i][2])

    else:

        CT.append(CT[i-1]+processes[i][2])

# Calculation Turn Around Time

for i in range(len(processes)):

    TAT.append(CT[i]-processes[i][1])

# Calculation Waiting Time

for i in range(len(processes)):

    WT.append(TAT[i]-processes[i][2])

table.add\_column("Process", justify="center")

table.add\_column("Arrival Time", justify="center")

table.add\_column("Burst Time", justify="center")

table.add\_column("Completion Time", justify="center")

table.add\_column("Turn Around Time", justify="center")

table.add\_column("Waiting Time", justify="center")

for i in range(len(processes)):

    table.add\_row(str(processes[i][0]), str(processes[i][1]), str(

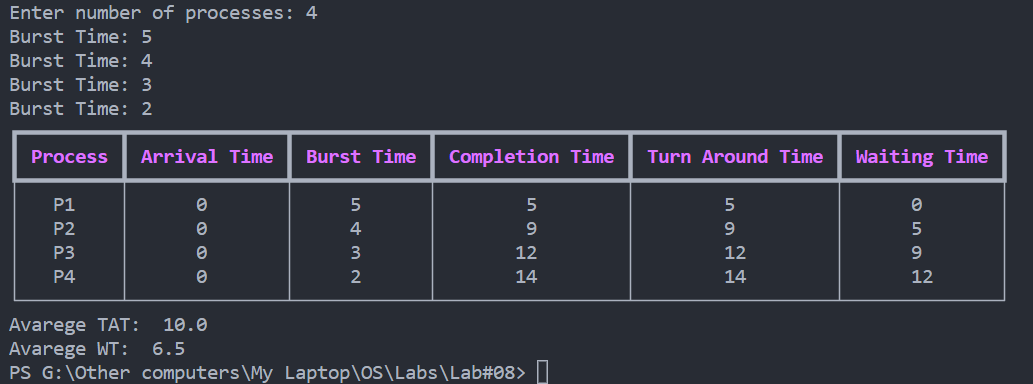
        processes[i][2]), str(CT[i]), str(TAT[i]), str(WT[i]))

console.print(table)

print("Avarege TAT: ", round(sum(TAT)/len(TAT), 2))

print("Avarege WT: ", round(sum(WT)/len(WT), 2))

Output:



# SJF (with arrival time 0):

import os

try:

    from rich.console import Console

    from rich.table import Table

except ModuleNotFoundError:

    os.system("pip install rich")

    from rich.console import Console

    from rich.table import Table

console = Console()

table = Table(show\_header=True, header\_style="bold magenta")

os.system("cls")

nprocess = int(input("Enter number of processes: "))

processes = []

CT = []

TAT = []

WT = []

for i in range(nprocess):

    b = int(input("Burst Time: "))

    processes.append(["P"+str(i+1), 0, b])

# sort According to burst time

processes.sort(key=lambda x: x[2])

# Calculting Completion time

for i in range(len(processes)):

    if i == 0:

        if processes[i][1] > 0:

            state\_idle = processes[i][1]

            CT.append(processes[i][2]+state\_idle)

        else:

            CT.append(processes[i][2])

    else:

        if CT[i-1] < processes[i][1]:

            idle\_state = processes[i][1] - CT[i-1]

            CT.append(CT[i-1]+processes[i][2]+idle\_state)

        else:

            CT.append(CT[i-1]+processes[i][2])

# Calculation Turn Around Time

for i in range(len(processes)):

    TAT.append(CT[i]-processes[i][1])

# Calculation Waiting Time

for i in range(len(processes)):

    WT.append(TAT[i]-processes[i][2])

table.add\_column("Process", justify="center")

table.add\_column("Arrival Time", justify="center")

table.add\_column("Burst Time", justify="center")

table.add\_column("Completion Time", justify="center")

table.add\_column("Turn Around Time", justify="center")

table.add\_column("Waiting Time", justify="center")

for i in range(len(processes)):

    table.add\_row(str(processes[i][0]), str(processes[i][1]), str(

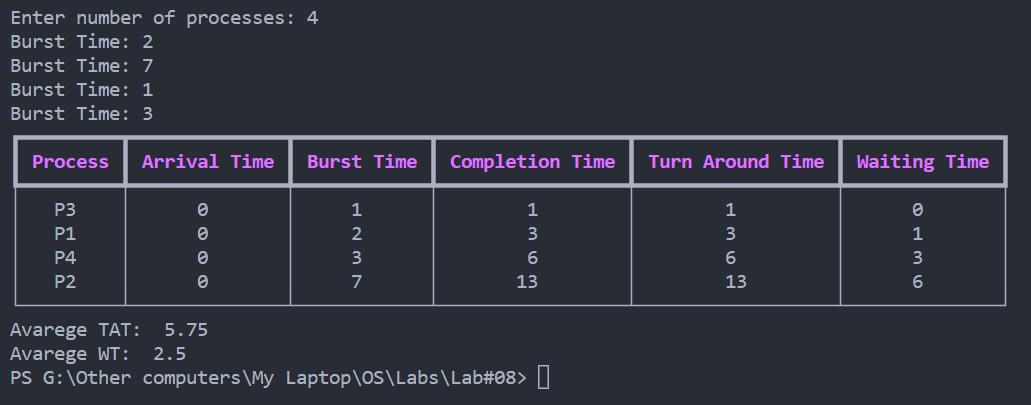
        processes[i][2]), str(CT[i]), str(TAT[i]), str(WT[i]))

console.print(table)

print("Avarege TAT: ", round(sum(TAT)/len(TAT), 2))

print("Avarege WT: ", round(sum(WT)/len(WT), 2))

Output:



# Modified FCFS for different arrival time and idleness:

import os

try:

    from rich.console import Console

    from rich.table import Table

except ModuleNotFoundError:

    os.system("pip install rich")

    from rich.console import Console

    from rich.table import Table

console = Console()

table = Table(show\_header=True, header\_style="bold magenta")

os.system("cls")

nprocess = int(input("Enter number of processes: "))

processes = []

CT = []

TAT = []

WT = []

for i in range(nprocess):

    a = int(input("Arrival time: "))

    b = int(input("Burst Time: "))

    processes.append(["P"+str(i+1), a, b])

# sort According to arrival time

processes.sort(key=lambda x: x[1])

# Calculting Completion time

for i in range(len(processes)):

    if i == 0:

        if processes[i][1] > 0:

            state\_idle = processes[i][1]

            CT.append(processes[i][2]+state\_idle)

        else:

            CT.append(processes[i][2])

    else:

        if CT[i-1] < processes[i][1]:

            idle\_state = processes[i][1] - CT[i-1]

            CT.append(CT[i-1]+processes[i][2]+idle\_state)

        else:

            CT.append(CT[i-1]+processes[i][2])

# Calculation Turn Around Time

for i in range(len(processes)):

    TAT.append(CT[i]-processes[i][1])

# Calculation Waiting Time

for i in range(len(processes)):

    WT.append(TAT[i]-processes[i][2])

table.add\_column("Process", justify="center")

table.add\_column("Arrival Time", justify="center")

table.add\_column("Burst Time", justify="center")

table.add\_column("Completion Time", justify="center")

table.add\_column("Turn Around Time", justify="center")

table.add\_column("Waiting Time", justify="center")

for i in range(len(processes)):

    table.add\_row(str(processes[i][0]), str(processes[i][1]), str(

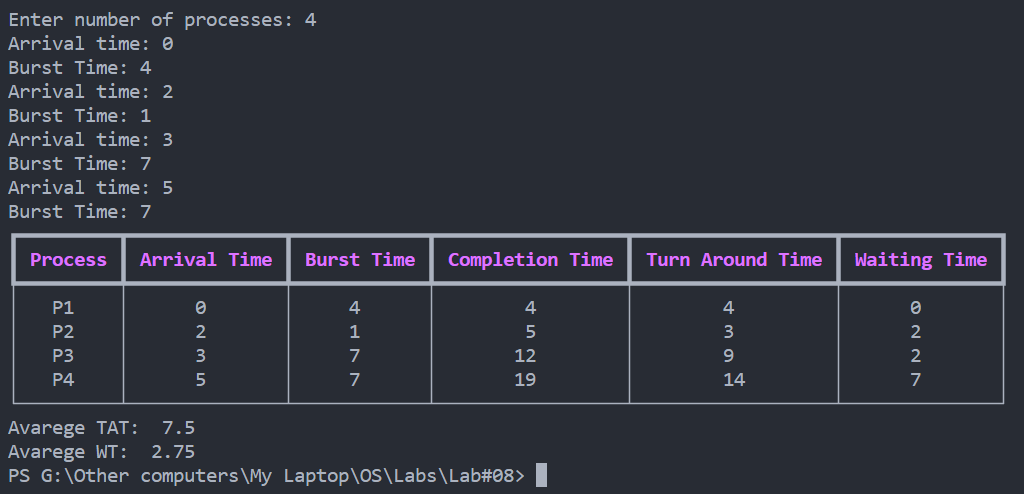
        processes[i][2]), str(CT[i]), str(TAT[i]), str(WT[i]))

console.print(table)

print("Avarege TAT: ", round(sum(TAT)/len(TAT), 2))

print("Avarege WT: ", round(sum(WT)/len(WT), 2))

Output:



# SJF (with different arrival time):

import os

try:

    from rich.console import Console

    from rich.table import Table

except ModuleNotFoundError:

    os.system("pip install rich")

    from rich.console import Console

    from rich.table import Table

console = Console()

table = Table(show\_header=True, header\_style="bold magenta")

os.system("cls")

nprocess = int(input("Enter number of processes: "))

processes = []

Sorted = []

CT = []

TAT = []

WT = []

for i in range(nprocess):

    a = int(input("Arrival time: "))

    b = int(input("Burst Time: "))

    processes.append(["P"+str(i+1), a, b])

n = len(processes)

# arranging

t = min(processes, key=lambda x: x[1])

t = t[1]

for i in range(n):

    reach\_pro = []

    flag = True

    while flag == True:

        for j in range(len(processes)):

            if processes[j][1] <= t:

                reach\_pro.append(processes[j])

        if len(reach\_pro) == 0:

            t += 1

        else:

            flag = False

    least\_bt = min(reach\_pro, key=lambda x: x[2])

    t = t + least\_bt[2]

    Sorted.append(least\_bt)

    processes.remove(least\_bt)

# Calculting Completion time

for i in range(len(Sorted)):

    if i == 0:

        if Sorted[i][1] > 0:

            state\_idle = Sorted[i][1]

            CT.append(Sorted[i][2]+state\_idle)

        else:

            CT.append(Sorted[i][2])

    else:

        if CT[i-1] < Sorted[i][1]:

            idle\_state = Sorted[i][1] - CT[i-1]

            CT.append(CT[i-1]+Sorted[i][2]+idle\_state)

        else:

            CT.append(CT[i-1]+Sorted[i][2])

# Calculation Turn Around Time

for i in range(len(Sorted)):

    TAT.append(CT[i]-Sorted[i][1])

# Calculation Waiting Time

for i in range(len(Sorted)):

    WT.append(TAT[i]-Sorted[i][2])

table.add\_column("Process", justify="center")

table.add\_column("Arrival Time", justify="center")

table.add\_column("Burst Time", justify="center")

table.add\_column("Completion Time", justify="center")

table.add\_column("Turn Around Time", justify="center")

table.add\_column("Waiting Time", justify="center")

for i in range(len(Sorted)):

    table.add\_row(str(Sorted[i][0]), str(Sorted[i][1]), str(

        Sorted[i][2]), str(CT[i]), str(TAT[i]), str(WT[i]))

console.print(table)

print("Avarege TAT: ", round(sum(TAT)/len(TAT), 2))

print("Avarege WT: ", round(sum(WT)/len(WT), 2))

Output:

