# Linux Command Line

อ.ดร.วัชรินทร์  สาระไชย

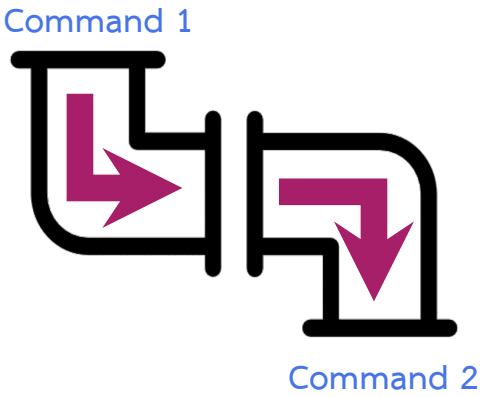
# การวัดผล

- วัดผลความเข้าใจคำสั่ง Linux command มาตรฐานโดยภาพรวมด้วยการให้นักศึกษาฝึกปฏิบัติ ประกอบไปด้วยคำสั่งมาตรฐานดังต่อไปนี้
  - การใช้ piping command
  - การใช้คำสั่ง tee command
  - การใช้คำสั่ง Xargs command
  - Creating File และ Directory
  - Copying Files และ Folders

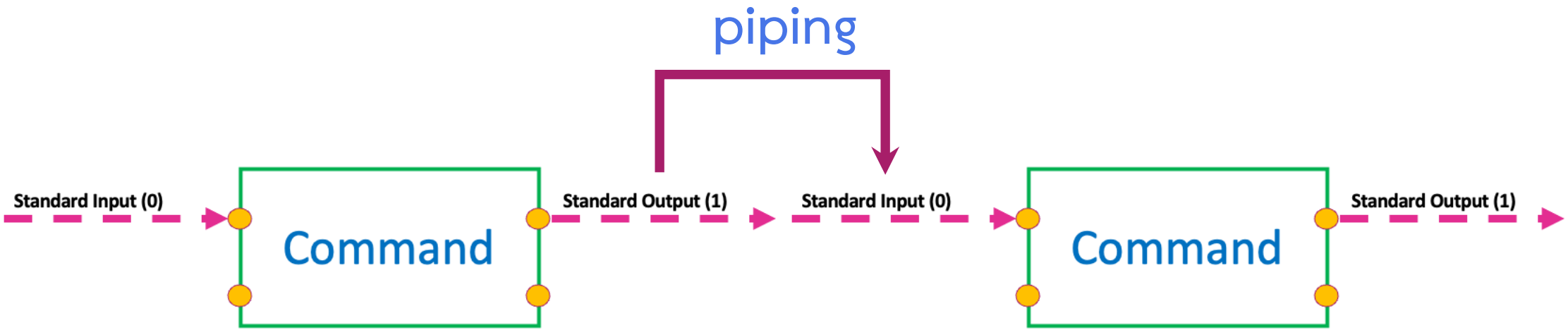
# Topic

- การใช้ piping command
- การใช้คำสั่ง tee command
- การใช้คำสั่ง Xargs command
- Creating File และ Directory
- Copying Files และ Folders

# Piping




>_ การส่ง output ของ command1 ไปเป็น input ของ command2



# Piping

- ตัวอย่างเช่นเมื่อเราพิมพ์คำสั่ง date แล้วเก็บผลลัพธ์ไว้ในไฟล์ date.txt

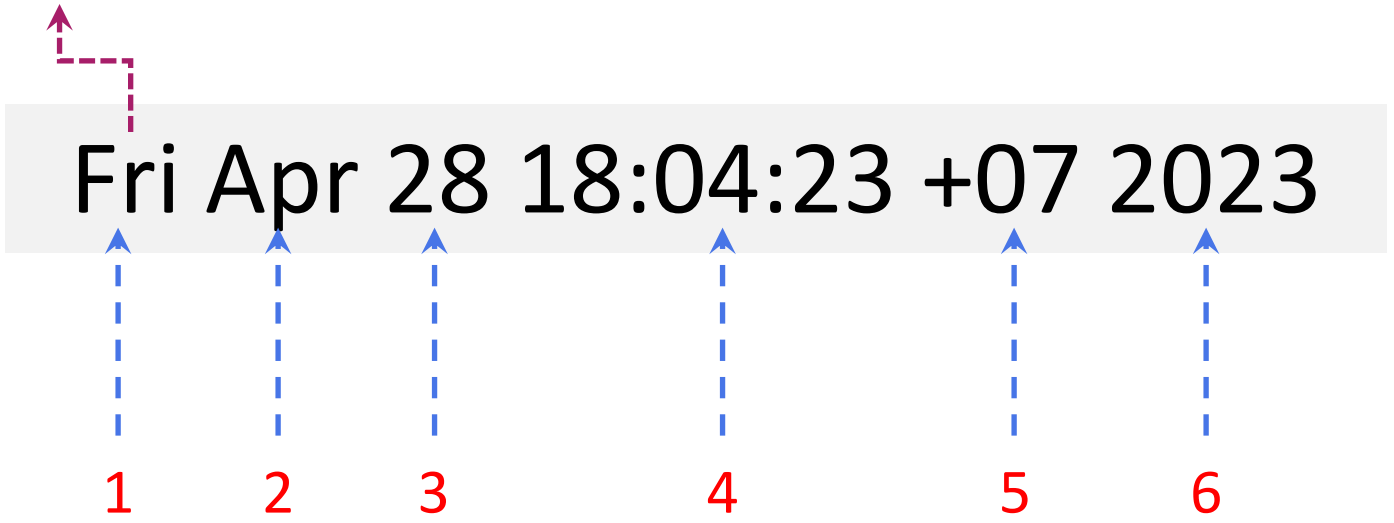
```
$ date
Fri Apr 28 18:04:23 +07 2023
$ date 1> date.txt
```

- เราสามารถตัดแต่ละคำของ “Fri Apr 28 18:04:23 +07 2023” ออกเป็นส่วน ๆ ได้โดยใช้คำสั่ง cut

# Piping

- คำสั่ง cut ทำหน้าที่ตัดส่วนของคำ โดยตัดแต่ละคำด้วยช่องว่างเราเรียกช่องว่าช่องว่านี้ว่า delimiter

```
itsci@itsci_mju_ac_th:~$ cut < date.txt --delimiter " " --fields 1
Fri
```




| | |
|---|---|
| 1 | Fri |
| 2 | Apr |
| 3 | 28 |
| 4 | 18:04:23 |
| 5 | +07 |
| 6 | 2023 |

# Piping

- Delimiter คือตัวอักษรคั่น จะเป็นตัวอักษรอะไรก็ได้ จากตัวอย่างนี้เราใช้ช่องว่าเป็น delimiter

# Piping

## คำสั่ง cut ทำหน้าที่ตัดส่วนของคำ

```
$ cut < date.txt --delimiter " " --fields 1
Fri
$ cut < date.txt --delimiter " " --fields 2
Apr
$ cut < date.txt --delimiter " " --fields 3
28
$ cut < date.txt --delimiter " " --fields 4
18:04:23
```

date.txt

```
Fri Apr 28 18:04:23 +07 2023
```

| | |
|---|---|
| 1 | Fri |
| 2 | Apr |
| 3 | 28 |
| 4 | 18:04:23 |
| 5 | +07 |
| 6 | 2023 |

# Piping

- อย่างไรก็ตามตัวอย่างคำสั่ง cut ก่อนหน้านี้อ่าน input จากไฟล์ date.txt
- หากเราต้องการส่ง output จากคำสั่ง date โดยตรงไปให้คำสั่ง cut จำเป็นต้องใช้วิธี piping

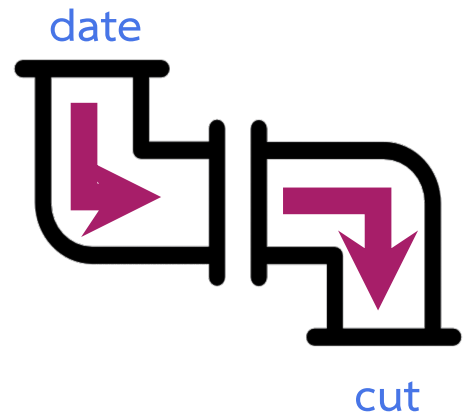
**จาก**

```
itsci@itsci_mju_ac_th:~$ cut < date.txt --delimiter " " --fields 1
Fri
```

**เป็น**

```
itsci@itsci_mju_ac_th:~$ date | cut --delimiter " " --fields 1
Fri
```

piping



# Piping

today.txt


- การ piping และเก็บผลลัพธ์ลงไฟล์ today.txt

```
$ date | cut --delimiter " " --fields 1 > today.txt
```

- เราสามารถวาง argument “> today.txt” ตำแหน่งไหนก็ได้หลังคำสั่ง cut

```
$ date | cut > today.txt --delimiter " " --fields 1
```

```
$ date | cut --delimiter " " > today.txt --fields 1
```

# Piping

- เราสามารถ piping ได้หลายต่อสามารถทำได้ เช่น

```
$ date | cut --delimiter " " --fields 1 | command -options args | ...
```

- หากเราส่ง output ไปแล้วจะไม่สามารถส่งต่อได้อีก เช่น

```
$ date > date.txt | cut --delimiter " " --fields 1
```

  - กรณีนี้จะไม่แสดง output ออกมา เนื่องจาก output สิ้นสุดที่ไฟล์ date.txt แล้ว

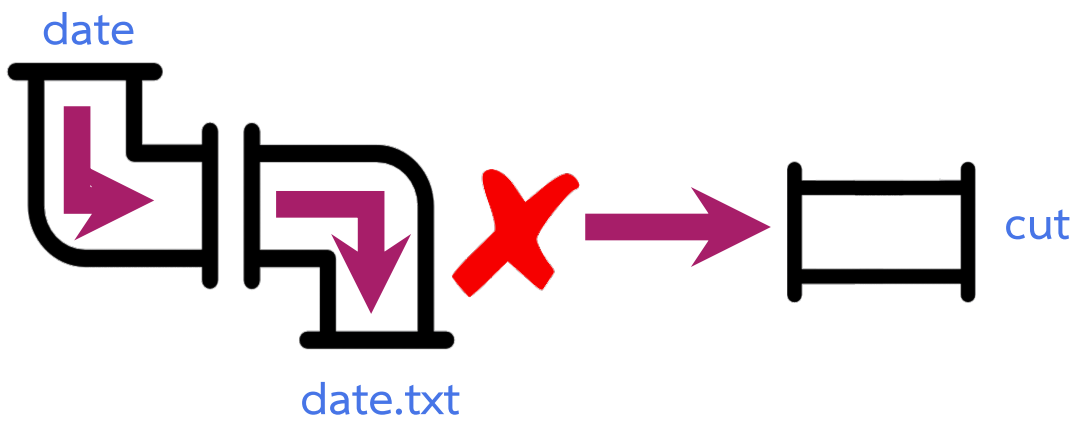
# Tee Command

- tee – สำเนา (duplicate) standard input
  - คำสั่ง tee เป็นเครื่องมือสำเนา standard input ไปยัง standard output
  - SYNOPSIS:
    - tee [-ai] [file ...]
    - options:
      - -a คือ ต่อท้ายผลลัพธ์ไปยังไฟล์แทนที่จะเขียนทับ
      - -i คือ ไม่สนใจ SIGINT signal
    - Operands
      - file คือ pathname ของ output file

# Tee Command

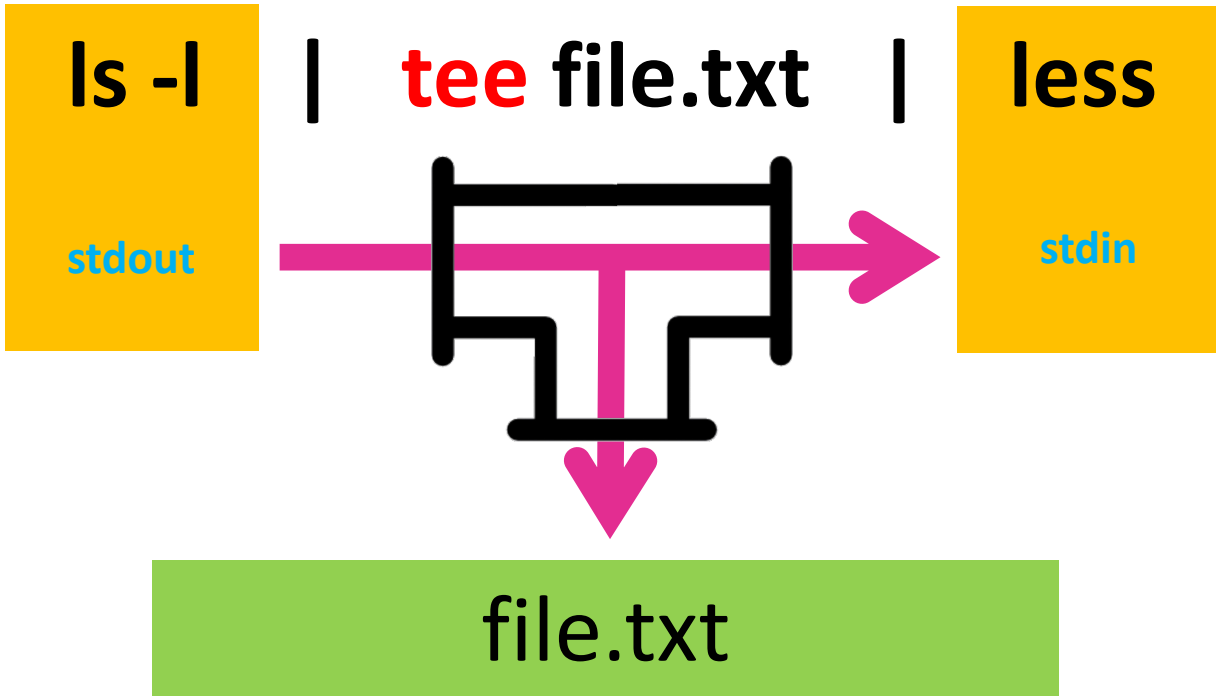
- จากปัญหาของคำสั่งนี้ข้อมูลจะสิ้นสุดลงและเก็บไว้ในไฟล์ “date.txt” จะไม่ส่งต่อไปยังคำสั่ง cut

```
$ date > date.txt | cut --delimiter " " --fields 1
```



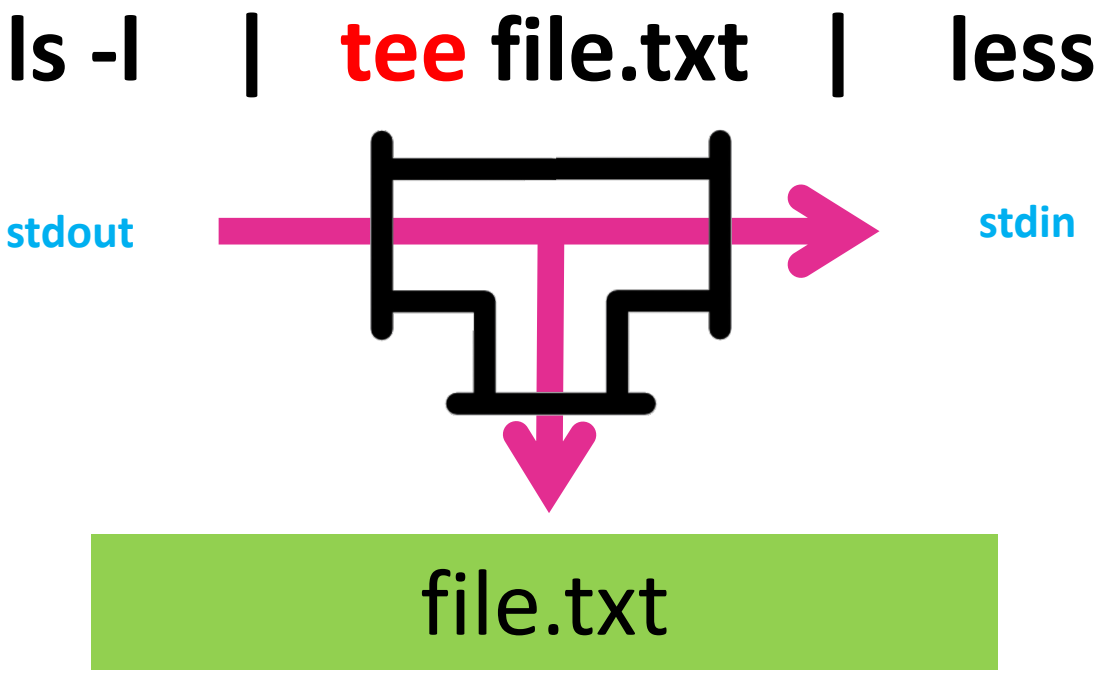
# Tee Command

ตัวอย่าง คำสั่ง tee



# Tee Command

## ตัวอย่าง คำสั่ง tee



# Tee Command

- จากคำสั่ง date ต่อไปนี้

```
$ date
Sun Apr 30 12:58:40 +07 2023
```

- เมื่อตัดคำออกด้วยคำสั่ง cut

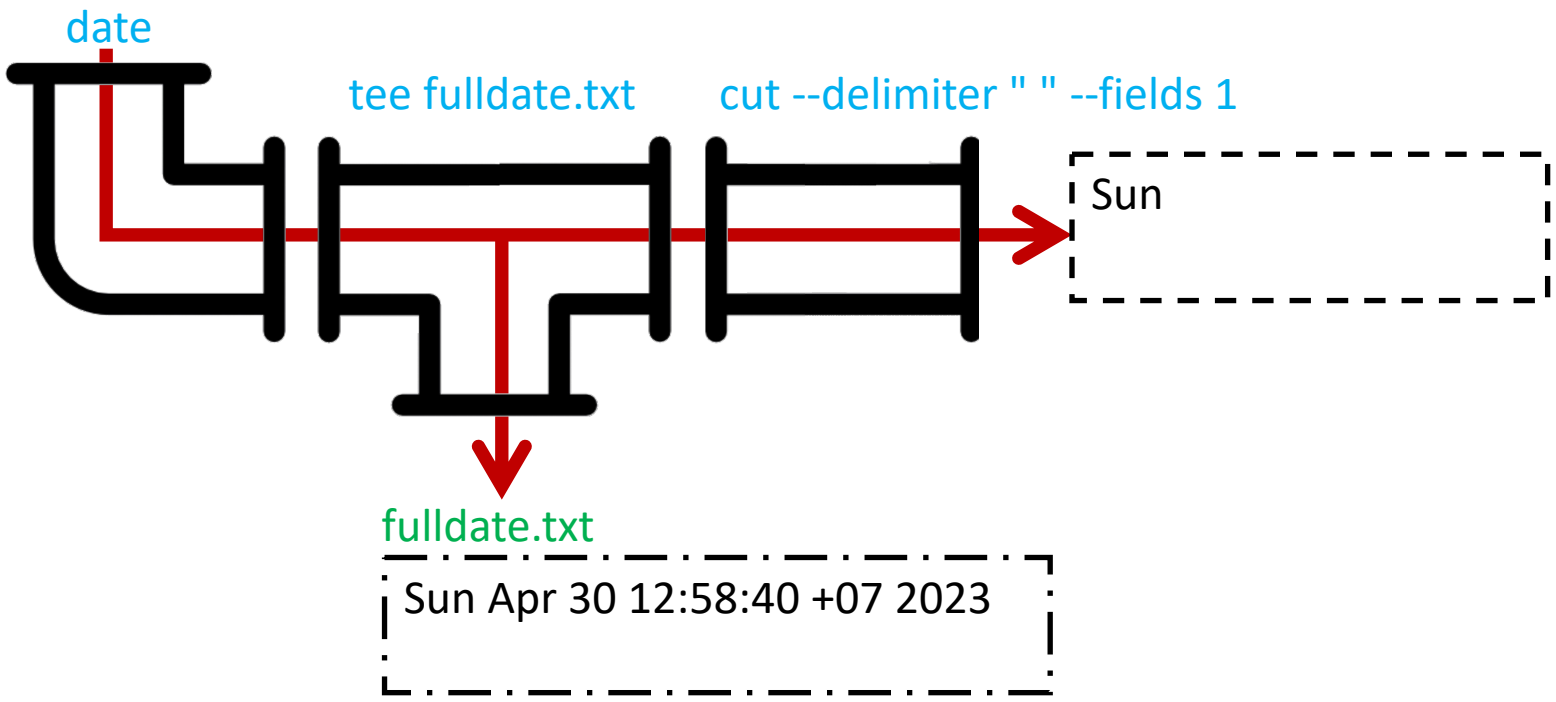
```
$ date | cut --delimiter " " --fields 1
Sun
```

- หากต้องการบันทึกวันที่ทั้งหมดไปพร้อมกันในคำสั่งเดียวจะต้องทำอย่างไร

```
$ date | tee fulldate.txt | cut --delimiter " " --fields 1
Sun
```

# Tee Command

```
$ date | tee fulldate.txt | cut --delimiter " " --fields 1
Sun
```



# Piping

- เมื่อข้อมูลส่งต่อไปจนถึงปลายทางด้วยเครื่องหมาย > แล้วจะไม่สามารถส่งต่อไปได้อีก เช่น

```
$ date | tee fulldate.txt | cut --delimiter " " --fields 1 > today.txt
```

```
$ date | tee fulldate.txt | cut --delimiter " " --fields 1 > today.txt | cmd
```

```
$ date | tee fulldate.txt | cut --delimiter " " --fields 1 | tee today.txt | cmd
```

# The Xargs Command

- คำสั่ง xargs ใช้สำหรับ build และประมวลผล command line จาก standard input
  - SYNOPSIS: [options] [command [initial-arguments]]
  - Example:

```
$ find /tmp –name core –type f -print | xargs /bin/rm -f
```

    - ค้นหาไฟล์ชื่อ core ใน /tmp และลบออก

# The Xargs Command

- คำสั่ง echo ไม่รับ Standard input แต่จะรับข้อมูลผ่าน arguments เท่านั้น

```
itsci@itsci_mju_ac_th:~$  echo Hello
Hello
```

# The Xargs Command

- หากส่งต่อข้อมูลในลักษณะ input ให้คำสั่ง echo จะไม่สามารถทำได้เช่น

```
itsci@itsci_mju_ac_th:~$  date | echo
```

- จะไม่มีผลลัพธ์ออกมา

# The Xargs Command

- ข้อแตกต่างระหว่าง arguments และ input stream คือ
  - Command line arguments เป็นส่วนหนึ่งของการเรียกใช้โปรแกรม และจะส่งข้อมูลให้โปรแกรมก่อนที่มันจะเริ่มทำงาน เป็นการป้อนข้อมูลให้โปรแกรมเพียงครั้งเดียว
  - Input stream จะส่งข้อมูลให้โปรแกรมหลังจากที่มันเริ่มทำงานแล้ว โดยโปรแกรมเมื่อเริ่มทำงานแล้วอาจโต้ตอบกับผู้ใช้เพื่อรับข้อมูลเพิ่มเติมได้อีก

# The Xargs Command

พิจารณาคำสั่งต่อไปนี้

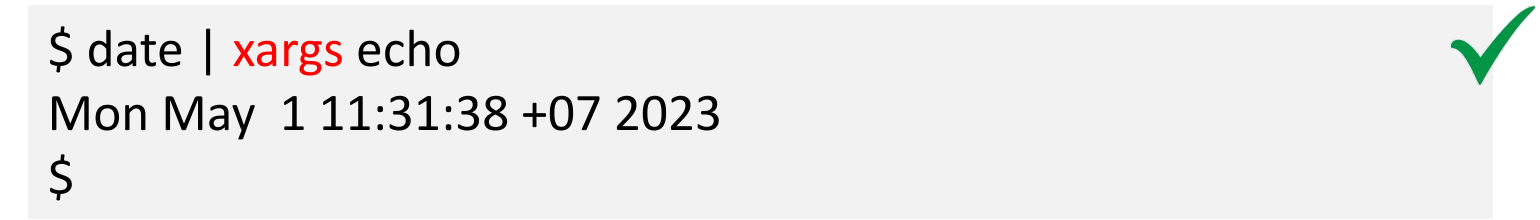
```
$ date
Mon May  1 11:31:38 +07 2023
$ echo “hello”
hello
$ date | echo

$
```

- คำสั่ง echo ไม่รับ standard input จะรับ input แบบ argument เท่านั้น
- จะทำอย่างไร ถ้าต้องการส่ง output stream ของคำสั่ง date ไปให้ echo ?

# The Xargs Command

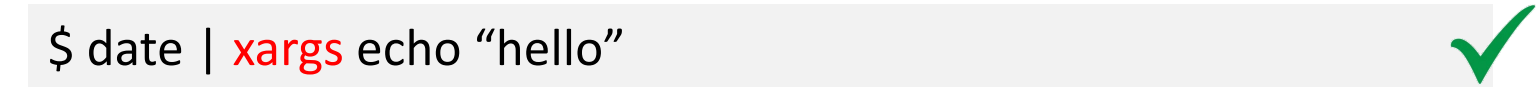
พิจารณาคำสั่งต่อไปนี้

```
$ date | xargs echo
Mon May  1 11:31:38 +07 2023
$
```



คำสั่ง date ส่ง output stream ไปให้ input stream ของคำสั่ง xargs ซึ่งจะแปลงเป็น input argument ส่งให้กับคำสั่ง echo

```
$ date | xargs echo “hello”
hello Mon May  1 11:31:38 +07 2023
$
```



# The Xargs Command

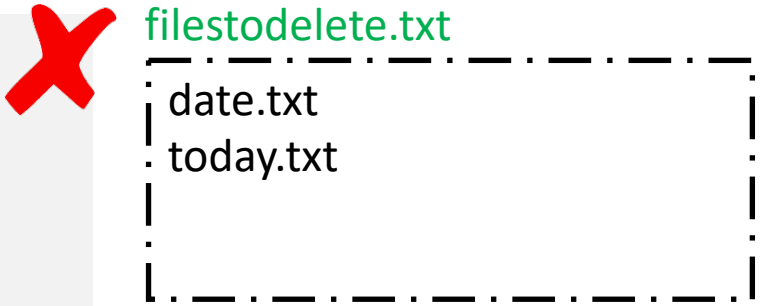
## คำสั่ง rm ใช้สำหรับลบไฟล์ตามต้องการเช่น

```
$ ls
date.txt  deleteme.txt  today.txt
$ rm deleteme.txt
date.txt  today.txt
```

```
$ rm deleteme.txt date.txt today.txt
```

<= ลบหลายไฟล์พร้อมกัน

```
$ cat filestodelete.txt
date.txt
today.txt
$ cat filetodelete.txt | rm  <= คำสั่ง rm ไม่ได้รับ input arguments
rm: missing operand
Try ‘rm –help’ for more information.
```




```
$ cat filetodelete.txt | xargs rm
```

<= ไฟล์ date.txt และ today.txt ถูกลบ

# Creating File and Directory

## องค์ประกอบของ Command Prompt

```
wsarachai@ITSCI:~$
```


## เราสามารถสอบถามว่า directory ปัจจุบันที่เราอยู่คืออะไรโดยใช้คำสั่ง pwd

```
wsarachai@ITSCI:~$ pwd
/home/wsarachai <= เครื่องหมาย “/” หน้าสุดแสดงว่าเป็น absolute path
```

# Command Pompt

- เราสามารถสอบถามว่ารายการไฟล์ต่าง ๆ ที่อยู่ใน directory ปัจจุบันโดยใช้คำสั่ง ls (ลองพิมพ์คำสั่ง man ls)
  - ตัวอย่างเช่น

```
wsarachai@ITSCI:~$ ls
wsarachai@ITSCI:~$ ls /home/wsarachai
wsarachai@ITSCI:~$ ls ~
wsarachai@ITSCI:~$ ls -F
wsarachai@ITSCI:~$ ls –F > list.txt
wsarachai@ITSCI:~$ ls –lh
```

แสดงผลลัพธ์เดียวกันทั้งสามคำสั่ง

# Wildcards

- เครื่องหมาย * คือ pattern ที่ใช้แทนชื่อของ file หรือ directory

```
wsarachai@ITSCI:~$ ls *    <= แสดงรายการ file และ directory ทั้งหมด
wsarachai@ITSCI:~$ ls Do*
wsarachai@ITSCI:~$ ls *.txt
```

- เครื่องหมาย ? แทนตัวอักษรอะไรก็ได้หนึ่งตัว

```
wsarachai@ITSCI:~$ ls file?.txt
wsarachai@ITSCI:~$ ls ??.txt
```

# Wildcards

- เครื่องหมาย [ ] แทนตัวอักษรหนึ่งตัวเหมือน ? แต่สามารถระบุช่วงข้อมูลได้

```
wsarachai@ITSCI:~$ ls file[123].txt      <= แสดงรายการ file1.txt, file2.txt, และ file3.txt
wsarachai@ITSCI:~$ ls file[0123456789].txt   <= แสดงรายการ file0.txt – file9.txt
wsarachai@ITSCI:~$ ls file[0-9].txt    <= แสดงรายการ file0.txt – file9.txt
wsarachai@ITSCI:~$ ls file[A-Z].txt    <= แสดงรายการ fileA.txt – fileZ.txt
wsarachai@ITSCI:~$ ls file[0-9][A-Z].txt  <= แสดงรายการ file0A.txt – file9Z.txt
wsarachai@ITSCI:~$ ls file[0-9ABC].txt  <= แสดงรายการ file0.txt หรือ fileA.txt และอื่น ๆ
```

# Wildcards

printf '%s\n' user{1,2,3}

- เครื่องหมาย { } แทนตัวอักษรหนึ่งตัวเหมือน ? แต่สามารถระบุช่วงข้อมูลได้

```
wsarachai@ITSCI:~$ echo 'user1\nuser2\nuser3\n'
user1\nuser2\nuser3\n
```

```
wsarachai@ITSCI:~$ echo –e 'user1\n user2\n user3\n'
user1
user2
user3
```

```
wsarachai@ITSCI:~$ printf '%s\n' user{1..3}
user1
user2
user3
```

# Creating Files and Directories

## การสร้าง blank file

```
wsarachai@ITSCI:~$ touch file1.txt
```

```
wsarachai@ITSCI:~$ touch /home/wsarachai/Document/file1.txt
```

## การสร้าง file พร้อมข้อมูล

```
wsarachai@ITSCI:~$ echo “hello” > file1.txt
```

# Creating Files and Directories

## การสร้าง directory ด้วยคำสั่ง mkdir

```
wsarachai@ITSCI:~$ mkdir folder
```

```
wsarachai@ITSCI:~$ mkdir ~/Pictures/holiday
```

## การสร้าง directory ซ้อนกันในทีเดียว

```
wsarachai@ITSCI:~$ mkdir bla/thing/keng
mkdir: cannot create directory ‘bla/thing/keng’: No such file or directory
```

<= bla ยังไม่มี

```
wsarachai@ITSCI:~$ mkdir -p bla/thing/keng
```

<= เพิ่ม options -p

# Creating Files and Directories

- คำสั่ง mkdir สามารถสร้างพร้อม ๆ กันหลาย directory ได้

```
wsarachai@ITSCI:~$ mkdir happy birthday
```

- การใช้ { } spread expression ในการสร้าง directory

```
wsarachai@ITSCI:~$ mkdir {project1,project2}_{A,B}   <= ระวังห้ามมีช่องว่างใน expression
wsarachai@ITSCI:~$ tree
.
├── project1_A
├── project1_B
├── project2_A
└── project2_B
```

# Creating Files and Directories

- คำสั่ง touch สามารถสร้างพร้อม ๆ กันหลาย file ได้เช่นกัน

```
wsarachai@ITSCI:~$ touch {project1,project2}_{A,B}
```

- การใช้ { } spread expression ในการสร้าง directory

```
wsarachai@ITSCI:~$ mkdir {project1,project2}_{A..Z}
wsarachai@ITSCI:~$ touch {project1,project2}_{A..Z}/file{1..100}.txt
wsarachai@ITSCI:~$ tree
.
├── project1_A
├── project1_B
├── ...
└── project2_Z
```

# Delete File and Directories

## คำสั่ง rm ใช้ลบ file

### ตัวอย่าง

```
wsarachai@ITSCI:~$ touch Documents/deleteme
wsarachai@ITSCI:~$ rm deleteme
rm: cannot remove 'deleteme': No such file or directory
```

```
wsarachai@ITSCI:~$ rm Documents/deleteme
```

# Delete File and Directories

- คำสั่ง rm ใช้ลบ directory จำเป็นต้องเพิ่ม options “-r” ด้วย
  - -r หมายถึงลบ directories และทุกอย่างทั้ง files, directories ที่ซ้อนต่อ ๆ กันไปใน directory (recursively)

```
wsarachai@ITSCI:~$ mkdir -p deleteme/folder{1..3}
wsarachai@ITSCI:~$ tree
.
└── deleteme
    ├── folder1
    ├── folder2
    └── folder3
wsarachai@ITSCI:~$ rm -r deleteme
```

# Delete File and Directories

- การใช้คำสั่ง rm -r มีความอันตรายอย่างมาก เช่น

```
wsarachai@ITSCI:~$ rm -r *    <= ลบทุก files และ directories ทันทีโดยไม่ถามยืนยัน
```

- ดังนั้นการเพิ่ม options “-i” คือให้ยืนยันก่อนลบทุก file หรือ directory ทุกครั้ง (prompt before every removal)
- ดังนั้นคำสั่งที่ปลอดภัยควรเป็นดังนี้

```
wsarachai@ITSCI:~$ rm -r -i *
wsarachai@ITSCI:~$ rm -ri *
```

} ทำงานเหมือนกัน

# Copying Files and Folders

- คำสั่ง cp ใช้สำหรับคัดลอง file
  - SYNOPSIS
    - `cp [-R [-H | -L | -P]] [-fi | -n] [-alpsvXx] source_file target_file`
    - `cp [-R [-H | -L | -P]] [-fi | -n] [-alpsvXx] source_file ... target_directory`
    - `cp [-f | -i | -n] [-alPpsvx] source_file target_file`
    - `cp [-f | -i | -n] [-alPpsvx] source_file ... target_directory`
  - รายละเอียดเพิ่มเติมให้พิมพ์คำสั่ง

```
wsarachai@ITSCI:~$ man cp
```

    - ตัวอย่างคำสั่ง

```
wsarachai@ITSCI:~$ cp file1.txt file2.txt
```

# Copying Files and Folders

## คำสั่ง cp ใช้สำหรับคัดลอง file

### ตัวอย่างการใช้คำสั่ง cp

```
wsarachai@ITSCI:~$ echo "Hello, this is file1.txt." > file1.txt
wsarachai@ITSCI:~$ cp file1.txt file2.txt
wsarachai@ITSCI:~$ mkdir dest
wsarachai@ITSCI:~$ cp file1.txt dest/
wsarachai@ITSCI:~$ cp file1.txt dest/file2.txt
```

```
wsarachai@ITSCI:~$ rm file*
wsarachai@ITSCI:~$ cp dest/* .
wsarachai@ITSCI:~$ cp dest/* ..
wsarachai@ITSCI:~$ cp dest/8 ../test/
wsarachai@ITSCI:~$ cp -r dest/ dest1/
```

# Moving and Renaming File and Directories

- คำสั่ง mv ใช้สำหรับย้ายหรือเปลี่ยนชื่อ file หรือ directory
  - SYNOPSIS
    - `mv [-f | -i | -n] [-hv] source target`
    - `mv [-f | -i | -n] [-v] source ... Directory`
  - รายละเอียดเพิ่มเติมให้พิมพ์คำสั่ง

```
wsarachai@ITSCI:~$ man mv
```

  - ตัวอย่างคำสั่ง

```
wsarachai@ITSCI:~$ mv file1.txt fileA.txt
```

# Copying Files and Folders

- คำสั่ง mv ใช้สำหรับย้ายหรือเปลี่ยนชื่อ file หรือ directory
  - ตัวอย่างการใช้คำสั่ง mv

```
wsarachai@ITSCI:~$ mv file1.txt fileA.txt
wsarachai@ITSCI:~$ mv dest dest-dir
wsarachai@ITSCI:~$ mv dest-dir/* .
wsarachai@ITSCI:~$ mv file* dest-dir/
wsarachai@ITSCI:~$ mv dest-dir/ ~/Documents/
```

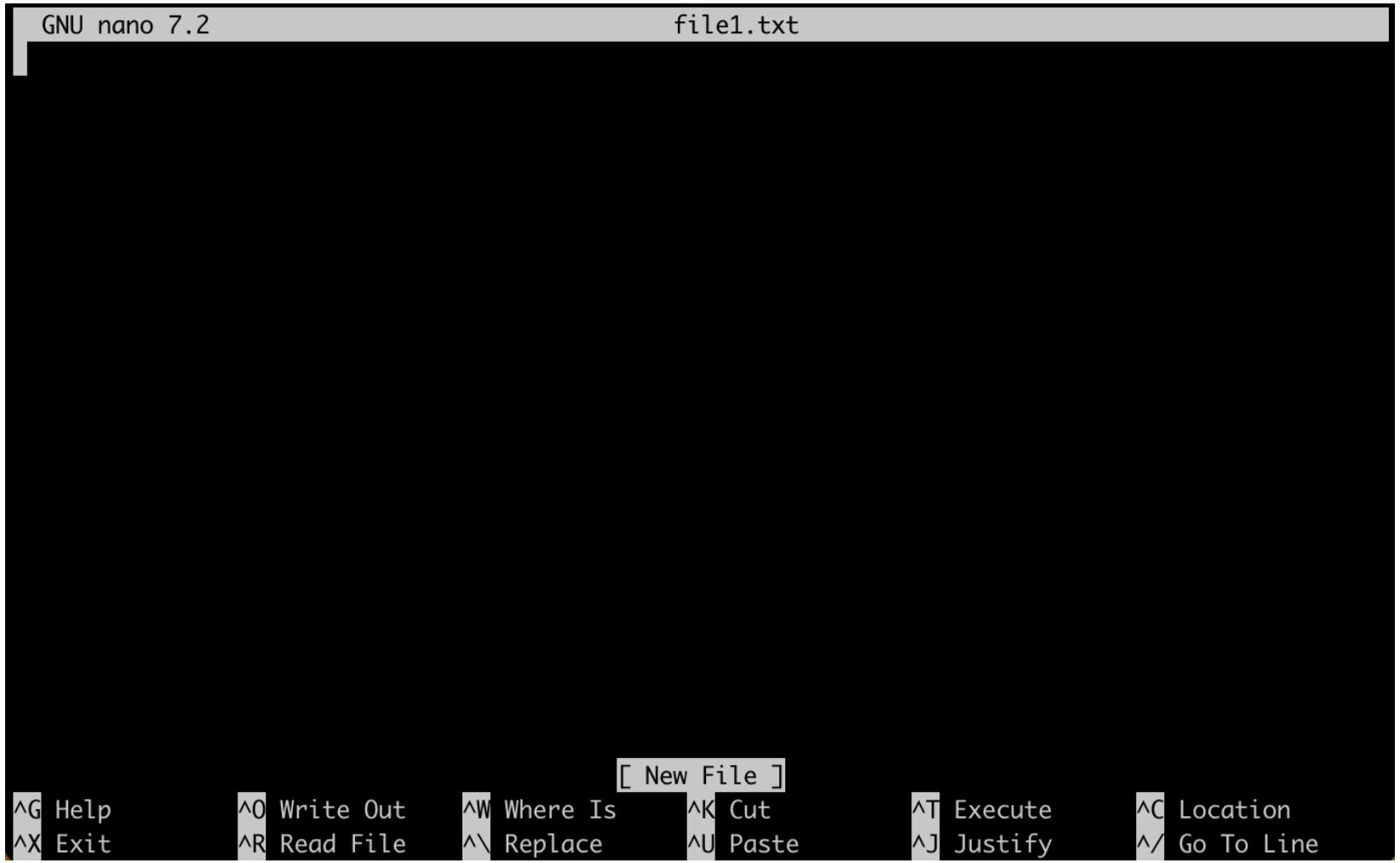
# Using nano Command Line Text Editor

- คำสั่ง nano ใช้สำหรับเป็น text editor แก้ไขไฟล์หรือสร้างไฟล์
  - Syntax
    - `nano [ options ] [ file ]`
  - รายละเอียดเพิ่มเติมให้พิมพ์คำสั่ง

```
wsarachai@ITSCI:~$ man nano
```

    - ตัวอย่างคำสั่ง

```
wsarachai@ITSCI:~$ nano file1.txt
```


GNU nano 7.2
file1.txt
[ New File ]
^G Help
^O Write Out
^W Where Is
^K Cut
^T Execute
^C Location
^X Exit
^R Read File
^\ Replace
^U Paste
^J Justify
^/ Go To Line

# Reference

- https://ubuntu.com/tutorials/command-line-for-beginners#1-overview
- https://www.youtube.com
- https://itsci.mju.ac.th/~watcharin/wordpress