

Terminal Adventure

Generated by Doxygen 1.8.13

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	LevelInformation Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Member Data Documentation	5
3.1.2.1	atk	5
3.1.2.2	count	6
3.1.2.3	def	6
3.1.2.4	hp	6
3.1.2.5	level	6
3.1.2.6	maxexp	6
3.1.2.7	mp	6
3.1.2.8	skills	6
3.2	MonsterInformation Struct Reference	7
3.2.1	Detailed Description	7
3.2.2	Member Data Documentation	7
3.2.2.1	atk	7
3.2.2.2	def	7
3.2.2.3	exp	7

3.2.2.4	hp	8
3.2.2.5	mp	8
3.2.2.6	name	8
3.2.2.7	skills	8
3.2.2.8	type	8
3.3	SkillInformation Struct Reference	8
3.3.1	Detailed Description	9
3.3.2	Member Data Documentation	9
3.3.2.1	dmg	9
3.3.2.2	mpcost	9
3.3.2.3	name	9
3.3.2.4	skilltype	9
3.3.2.5	type	9
3.4	StageInformation Struct Reference	10
3.4.1	Detailed Description	10
3.4.2	Member Data Documentation	10
3.4.2.1	monsters	10
3.4.2.2	number	10
3.5	UserInfoNode Struct Reference	10
3.5.1	Detailed Description	11
3.5.2	Member Data Documentation	11
3.5.2.1	loggedin	11
3.5.2.2	next	11
3.5.2.3	user	11
3.6	UserInfoInformation Struct Reference	11
3.6.1	Detailed Description	12
3.6.2	Member Data Documentation	12
3.6.2.1	curExp	12
3.6.2.2	curHP	12
3.6.2.3	curMP	12
3.6.2.4	level	12
3.6.2.5	password	13
3.6.2.6	stage	13
3.6.2.7	username	13

4 File Documentation	15
4.1 client.c File Reference	15
4.1.1 Macro Definition Documentation	16
4.1.1.1 BUFFER_LEN	16
4.1.2 Function Documentation	16
4.1.2.1 main()	16
4.1.3 Variable Documentation	16
4.1.3.1 ch	16
4.1.3.2 servaddr	16
4.1.3.3 sockfd	16
4.1.3.4 temp	16
4.1.3.5 user	17
4.2 game/game.c File Reference	17
4.2.1 Function Documentation	18
4.2.1.1 campaign()	18
4.2.1.2 damageCalculationMonster()	18
4.2.1.3 damageCalculationPlayer()	19
4.2.1.4 gameoverChoice()	19
4.2.1.5 monsterAI()	19
4.2.1.6 printMonsterLog()	20
4.2.1.7 printUserLog()	20
4.2.1.8 stageoverChoice()	20
4.2.2 Variable Documentation	21
4.2.2.1 curDmg	21
4.2.2.2 levels	21
4.2.2.3 monsterCurHP	21
4.2.2.4 monsterCurMP	21
4.2.2.5 monsters	21
4.2.2.6 monsterSkill	21
4.2.2.7 skills	21

4.2.2.8	sockfd	22
4.2.2.9	stages	22
4.2.2.10	user	22
4.2.2.11	userCurExp	22
4.2.2.12	userCurHP	22
4.2.2.13	userCurLevel	22
4.2.2.14	userCurMP	22
4.2.2.15	userCurStage	22
4.2.2.16	userStageHP	23
4.2.2.17	userStageMP	23
4.3	game/game.h File Reference	23
4.3.1	Macro Definition Documentation	24
4.3.1.1	BLU	24
4.3.1.2	BUF_SIZE	24
4.3.1.3	CYN	24
4.3.1.4	GRN	24
4.3.1.5	MAG	24
4.3.1.6	RED	24
4.3.1.7	RESET	24
4.3.1.8	WHT	25
4.3.1.9	YEL	25
4.3.2	Function Documentation	25
4.3.2.1	campaign()	25
4.3.2.2	damageCalculationMonster()	25
4.3.2.3	damageCalculationPlayer()	25
4.3.2.4	gameoverChoice()	26
4.3.2.5	monsterAI()	26
4.3.2.6	printMonsterLog()	26
4.3.2.7	printUserLog()	27
4.3.2.8	stageoverChoice()	27

4.4	game/gamemaster.c File Reference	27
4.4.1	Function Documentation	28
4.4.1.1	loadUserInfo()	28
4.5	game/gamemaster.h File Reference	28
4.5.1	Function Documentation	28
4.5.1.1	loadUserInfo()	28
4.5.2	Variable Documentation	29
4.5.2.1	root	29
4.6	helper/helper.c File Reference	29
4.6.1	Function Documentation	29
4.6.1.1	userHighScoreFormat()	29
4.7	helper/helper.h File Reference	29
4.7.1	Macro Definition Documentation	30
4.7.1.1	MAX_STAGE_IN_STRING	30
4.7.2	Function Documentation	30
4.7.2.1	userHighScoreFormat()	30
4.8	helper/mysocket.c File Reference	30
4.8.1	Function Documentation	31
4.8.1.1	configAddress()	31
4.8.1.2	die()	31
4.8.1.3	getPort()	32
4.9	helper/mysocket.h File Reference	32
4.9.1	Function Documentation	32
4.9.1.1	configAddress()	32
4.9.1.2	die()	33
4.9.1.3	getPort()	33
4.10	interface/clientfunc.c File Reference	33
4.10.1	Function Documentation	34
4.10.1.1	fetchHighScore()	34
4.10.1.2	fetchPlayerData()	34

4.10.1.3	login()	35
4.10.1.4	playMenu()	35
4.10.1.5	registerFunc()	35
4.10.1.6	updateUserInfo()	36
4.10.2	Variable Documentation	36
4.10.2.1	servaddr	36
4.10.2.2	user	36
4.11	interface/clientfunc.h File Reference	36
4.11.1	Macro Definition Documentation	37
4.11.1.1	BUFFER_LEN	37
4.11.2	Function Documentation	37
4.11.2.1	fetchHighScore()	37
4.11.2.2	fetchPlayerData()	38
4.11.2.3	login()	38
4.11.2.4	playMenu()	38
4.11.2.5	registerFunc()	38
4.11.2.6	updateUserInfo()	39
4.12	interface/serverfunc.c File Reference	39
4.12.1	Function Documentation	39
4.12.1.1	addUser()	39
4.12.1.2	handleRequest()	40
4.12.1.3	login()	40
4.12.1.4	sendHighScore()	41
4.12.1.5	sendPlayerInfo()	41
4.12.1.6	updatePlayerInfo()	41
4.13	interface/serverfunc.h File Reference	42
4.13.1	Macro Definition Documentation	43
4.13.1.1	BUFFER_LEN	43
4.13.2	Function Documentation	43
4.13.2.1	addUser()	43

4.13.2.2	handleRequest()	43
4.13.2.3	login()	44
4.13.2.4	sendHighScore()	44
4.13.2.5	sendPlayerInfo()	44
4.13.2.6	updatePlayerInfo()	45
4.14	server.c File Reference	45
4.14.1	Macro Definition Documentation	46
4.14.1.1	MAX_CLIENT	46
4.14.2	Function Documentation	46
4.14.2.1	main()	46
4.14.3	Variable Documentation	46
4.14.3.1	root	46
4.15	struct/level.c File Reference	46
4.15.1	Function Documentation	47
4.15.1.1	loadLevelInfo()	47
4.15.2	Variable Documentation	47
4.15.2.1	levels	47
4.15.2.2	skills	47
4.16	struct/level.h File Reference	47
4.16.1	Macro Definition Documentation	48
4.16.1.1	MAX_LEVEL	48
4.16.2	Typedef Documentation	48
4.16.2.1	LevelInfo	48
4.16.3	Function Documentation	48
4.16.3.1	loadLevelInfo()	48
4.17	struct/monster.c File Reference	49
4.17.1	Function Documentation	49
4.17.1.1	loadMonsterInfo()	49
4.17.2	Variable Documentation	49
4.17.2.1	monsters	49

4.17.2.2	skills	49
4.18	struct/monster.h File Reference	50
4.18.1	Macro Definition Documentation	50
4.18.1.1	MAX_MONSTER	50
4.18.2	Typedef Documentation	50
4.18.2.1	MonsterInfo	50
4.18.3	Function Documentation	51
4.18.3.1	loadMonsterInfo()	51
4.19	struct/skill.c File Reference	51
4.19.1	Variable Documentation	51
4.19.1.1	skills	51
4.20	struct/skill.h File Reference	52
4.20.1	Macro Definition Documentation	53
4.20.1.1	SKILL_COUNT	53
4.20.1.2	STRING_LEN	53
4.20.2	Typedef Documentation	53
4.20.2.1	SkillInfo	53
4.20.3	Enumeration Type Documentation	53
4.20.3.1	Skill	53
4.20.3.2	Type	54
4.21	struct/stage.c File Reference	55
4.21.1	Function Documentation	55
4.21.1.1	loadStageInfo()	55
4.21.2	Variable Documentation	55
4.21.2.1	monsters	55
4.21.2.2	stages	55
4.22	struct/stage.h File Reference	56
4.22.1	Macro Definition Documentation	56
4.22.1.1	MAX_STAGE	56
4.22.2	Typedef Documentation	56

4.22.2.1	StageInfo	56
4.22.3	Function Documentation	57
4.22.3.1	loadStageInfo()	57
4.23	struct/user.c File Reference	57
4.23.1	Function Documentation	57
4.23.1.1	findUser()	57
4.23.1.2	freeList()	58
4.23.1.3	initUserInfo()	58
4.23.1.4	insertNode()	58
4.23.1.5	makeNewNode()	60
4.23.1.6	printList()	60
4.23.1.7	sortUserlist()	60
4.23.1.8	writeUserData()	61
4.24	struct/user.h File Reference	61
4.24.1	Detailed Description	62
4.24.2	Macro Definition Documentation	62
4.24.2.1	STRING_LEN	62
4.24.3	Typedef Documentation	62
4.24.3.1	UserInfo	62
4.24.3.2	UserNode	63
4.24.4	Function Documentation	63
4.24.4.1	findUser()	63
4.24.4.2	freeList()	63
4.24.4.3	initUserInfo()	63
4.24.4.4	insertNode()	64
4.24.4.5	makeNewNode()	64
4.24.4.6	printList()	65
4.24.4.7	sortUserlist()	65
4.24.4.8	writeUserData()	65

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

LevelInformation		
LevelInformation	Structure	5
MonsterInformation		
MonsterInformation	structure	7
SkillInformation		
SkillInformation	Structure	8
StageInformation		
StageInformation	structure	10
UserInfoNode		
UserInfoNode	structure	10
UserInformation		
UserInformation	structure	11

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

client.c	15
server.c	45
game/ game.c	17
game/ game.h	23
game/ gamemaster.c	27
game/ gamemaster.h	28
helper/ helper.c	29
helper/ helper.h	29
helper/ mysocket.c	30
helper/ mysocket.h	32
interface/ clientfunc.c	33
interface/ clientfunc.h	36
interface/ serverfunc.c	39
interface/ serverfunc.h	42
struct/ level.c	46
struct/ level.h	47
struct/ monster.c	49
struct/ monster.h	50
struct/ skill.c	51
struct/ skill.h	52
struct/ stage.c	55
struct/ stage.h	56
struct/ user.c	57
struct/ user.h	
File containing User structure and usage	61

Chapter 3

Class Documentation

3.1 LevelInformation Struct Reference

[LevelInformation](#) Structure.

```
#include <level.h>
```

Public Attributes

- int [level](#)
- int [hp](#)
- int [mp](#)
- int [atk](#)
- int [def](#)
- int [maxexp](#)
- int [count](#)
- [SkillInfo](#) [skills](#) [20]

3.1.1 Detailed Description

[LevelInformation](#) Structure.

Structure to store Player Level's Information

3.1.2 Member Data Documentation

3.1.2.1 atk

```
int LevelInformation::atk
```

Attack

3.1.2.2 count

```
int LevelInformation::count
```

Skill count

3.1.2.3 def

```
int LevelInformation::def
```

Defend

3.1.2.4 hp

```
int LevelInformation::hp
```

HP

3.1.2.5 level

```
int LevelInformation::level
```

Level

3.1.2.6 maxexp

```
int LevelInformation::maxexp
```

Max Exp

3.1.2.7 mp

```
int LevelInformation::mp
```

MP

3.1.2.8 skills

```
SkillInfo LevelInformation::skills[20]
```

The documentation for this struct was generated from the following file:

- [struct/level.h](#)

3.2 MonsterInformation Struct Reference

[MonsterInformation](#) structure.

```
#include <monster.h>
```

Public Attributes

- char [name](#) [[STRING_LEN](#)]
- int [hp](#)
- int [mp](#)
- int [atk](#)
- int [def](#)
- int [exp](#)
- [Type](#) [type](#)
- [SkillInfo](#) [skills](#) [4]

3.2.1 Detailed Description

[MonsterInformation](#) structure.

Structure to store monster's information

3.2.2 Member Data Documentation

3.2.2.1 atk

```
int MonsterInformation::atk
```

Monster attack

3.2.2.2 def

```
int MonsterInformation::def
```

Monster defend

3.2.2.3 exp

```
int MonsterInformation::exp
```

EXP ammount earned on Monster defeat

3.2.2.4 hp

```
int MonsterInformation::hp
```

Monster hp

3.2.2.5 mp

```
int MonsterInformation::mp
```

Monster mp

3.2.2.6 name

```
char MonsterInformation::name[STRING\_LEN]
```

Monster name

3.2.2.7 skills

```
SkillInfo MonsterInformation::skills[4]
```

Monster skill

3.2.2.8 type

```
Type MonsterInformation::type
```

Monster Type

The documentation for this struct was generated from the following file:

- struct/[monster.h](#)

3.3 SkillInformation Struct Reference

[SkillInformation](#) Structure.

```
#include <skill.h>
```

Public Attributes

- [Skill](#) skilltype
- char [name](#) [[STRING_LEN](#)]
- int [dmg](#)
- int [mpcost](#)
- [Type](#) type

3.3.1 Detailed Description

[SkillInformation](#) Structure.

Structure to store Skill's Information

3.3.2 Member Data Documentation

3.3.2.1 dmg

```
int SkillInformation::dmg
```

Skill damage

3.3.2.2 mpcost

```
int SkillInformation::mpcost
```

Skill MP cost

3.3.2.3 name

```
char SkillInformation::name[STRING\_LEN]
```

Skill name

3.3.2.4 skilltype

```
Skill SkillInformation::skilltype
```

Skill name type

3.3.2.5 type

```
Type SkillInformation::type
```

Skill action type

The documentation for this struct was generated from the following file:

- struct/[skill.h](#)

3.4 StageInformation Struct Reference

[StageInformation](#) structure.

```
#include <stage.h>
```

Public Attributes

- int [number](#)
- [MonsterInfo](#) [monsters](#) [10]

3.4.1 Detailed Description

[StageInformation](#) structure.

Structure to store stage's information

3.4.2 Member Data Documentation

3.4.2.1 monsters

[MonsterInfo](#) [StageInformation::monsters](#)[10]

Stage's monsters

3.4.2.2 number

int [StageInformation::number](#)

Stage number

The documentation for this struct was generated from the following file:

- struct/[stage.h](#)

3.5 UserInfoNode Struct Reference

[UserInfoNode](#) structure.

```
#include <user.h>
```

Public Attributes

- [UserInfo](#) `user`
- `int` `loggedin`
- `struct` [UserInfoNode](#) * `next`

3.5.1 Detailed Description

[UserInfoNode](#) structure.

Structure to store user's information as linked list

3.5.2 Member Data Documentation

3.5.2.1 `loggedin`

```
int UserInfoNode::loggedin
```

Login status

3.5.2.2 `next`

```
struct UserInfoNode* UserInfoNode::next
```

Next user

3.5.2.3 `user`

```
UserInfo UserInfoNode::user
```

UserInfo

The documentation for this struct was generated from the following file:

- `struct`/[user.h](#)

3.6 UserInformation Struct Reference

[UserInformation](#) structure.

```
#include <user.h>
```

Public Attributes

- char `username` [`STRING_LEN`]
- char `password` [`STRING_LEN`]
- int `level`
- int `curExp`
- int `curHP`
- int `curMP`
- int `stage`

3.6.1 Detailed Description

`UserInfo` structure.

Structure to store user's information

3.6.2 Member Data Documentation

3.6.2.1 `curExp`

```
int UserInfo::curExp
```

Current EXP

3.6.2.2 `curHP`

```
int UserInfo::curHP
```

Current HP

3.6.2.3 `curMP`

```
int UserInfo::curMP
```

Current MP

3.6.2.4 `level`

```
int UserInfo::level
```

Level

3.6.2.5 password

```
char UserInformation::password[STRING_LEN]
```

Password

3.6.2.6 stage

```
int UserInformation::stage
```

Current stage

3.6.2.7 username

```
char UserInformation::username[STRING_LEN]
```

Username

The documentation for this struct was generated from the following file:

- [struct/user.h](#)

Chapter 4

File Documentation

4.1 client.c File Reference

```
#include <sys/types.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include "struct/skill.h"
#include "struct/monster.h"
#include "struct/level.h"
#include "struct/stage.h"
#include "struct/user.h"
#include "game/game.h"
#include "helper/mysocket.h"
#include "interface/clientfunc.h"
```

Macros

- `#define BUFFER_LEN 1024`

Functions

- `int main (int argc, char const *argv[])`

Variables

- `char ch`
- `char temp`
- `UserInfo user`
- `struct sockaddr_in servaddr`
- `int sockfd`

4.1.1 Macro Definition Documentation

4.1.1.1 BUFFER_LEN

```
#define BUFFER_LEN 1024
```

4.1.2 Function Documentation

4.1.2.1 main()

```
int main (
    int argc,
    char const * argv[] )
```

4.1.3 Variable Documentation

4.1.3.1 ch

```
char ch
```

4.1.3.2 servaddr

```
struct sockaddr_in servaddr
```

4.1.3.3 sockfd

```
int sockfd
```

4.1.3.4 temp

```
char temp
```

4.1.3.5 user

`UserInfo` user

4.2 game/game.c File Reference

```
#include <sys/types.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include "../struct/skill.h"
#include "../struct/monster.h"
#include "../struct/level.h"
#include "../struct/stage.h"
#include "../struct/user.h"
#include "game.h"
#include "../interface/clientfunc.h"
```

Functions

- void `campaign` ()
PVE mode.
- int `damageCalculationPlayer` (`SkillInfo` skill, `LevelInfo` player, `MonsterInfo` monster)
Player's damage calculation.
- int `damageCalculationMonster` (`SkillInfo` skill, `LevelInfo` player, `MonsterInfo` monster)
Monster's damage calculation.
- `SkillInfo` `monsterAI` (`MonsterInfo` monster, int curHP)
Monster's behavior.
- char `gameoverChoice` ()
Game over menu.
- char `stageoverChoice` ()
Game over menu.
- void `printUserLog` (char `user`[], char skill[], int dmg, `Type` type)
Print User log.
- void `printMonsterLog` (char `user`[], char skill[], int dmg, `Type` type)
Print Monster log.

Variables

- [SkillInfo](#) skills []
- [LevelInfo](#) levels []
- [MonsterInfo](#) monsters []
- [StageInfo](#) stages []
- int userCurHP
- int userCurMP
- int userCurLevel
- int userCurExp
- int userCurStage
- int monsterCurHP
- int monsterCurMP
- int curDmg
- int userStageHP
- int userStageMP
- [SkillInfo](#) monsterSkill
- [UserInfo](#) user
- int sockfd

4.2.1 Function Documentation

4.2.1.1 campaign()

```
void campaign ( )
```

PVE mode.

4.2.1.2 damageCalculationMonster()

```
int damageCalculationMonster (
    SkillInfo skill,
    LevelInfo player,
    MonsterInfo monster )
```

Monster's damage calculation.

Parameters

<i>SkillInfo</i>	Monster skill
<i>LevelInfo</i>	Player's level info
<i>MonsterInfo</i>	Monster information

4.2.1.3 damageCalculationPlayer()

```
int damageCalculationPlayer (
    SkillInfo skill,
    LevelInfo player,
    MonsterInfo monster )
```

Player's damage calculation.

Parameters

<i>SkillInfo</i>	Player skill
<i>LevelInfo</i>	Player's level info
<i>MonsterInfo</i>	Monster information

4.2.1.4 gameoverChoice()

```
char gameoverChoice ( )
```

Game over menu.

Returns

Player choice

4.2.1.5 monsterAI()

```
SkillInfo monsterAI (
    MonsterInfo monster,
    int curHP )
```

Monster's behavior.

Parameters

<i>MonsterInfo</i>	Monster information
<i>int</i>	Monster current HP

Returns

Monster Skill

4.2.1.6 printMonsterLog()

```
void printMonsterLog (
    char monster[],
    char skill[],
    int dmg,
    Type type )
```

Print Monster log.

Parameters

<i>String</i>	Monster name
<i>String</i>	Skill name
<i>int</i>	Damage
<i>Type</i>	Skill type

4.2.1.7 printUserLog()

```
void printUserLog (
    char user[],
    char skill[],
    int dmg,
    Type type )
```

Print User log.

Parameters

<i>String</i>	Username
<i>String</i>	Skill name
<i>int</i>	Damage
<i>Type</i>	Skill type

4.2.1.8 stageoverChoice()

```
char stageoverChoice ( )
```

Game over menu.

Returns

Player choice

4.2.2 Variable Documentation

4.2.2.1 curDmg

```
int curDmg
```

4.2.2.2 levels

```
LevelInfo levels[ ]
```

4.2.2.3 monsterCurHP

```
int monsterCurHP
```

4.2.2.4 monsterCurMP

```
int monsterCurMP
```

4.2.2.5 monsters

```
MonsterInfo monsters[ ]
```

4.2.2.6 monsterSkill

```
SkillInfo monsterSkill
```

4.2.2.7 skills

```
SkillInfo skills[ ]
```

4.2.2.8 sockfd

```
int sockfd
```

4.2.2.9 stages

```
StageInfo stages[ ]
```

4.2.2.10 user

```
UserInfo user
```

4.2.2.11 userCurExp

```
int userCurExp
```

4.2.2.12 userCurHP

```
int userCurHP
```

4.2.2.13 userCurLevel

```
int userCurLevel
```

4.2.2.14 userCurMP

```
int userCurMP
```

4.2.2.15 userCurStage

```
int userCurStage
```

4.2.2.16 userStageHP

```
int userStageHP
```

4.2.2.17 userStageMP

```
int userStageMP
```

4.3 game/game.h File Reference

```
#include "../struct/monster.h"
#include "../struct/skill.h"
#include "../struct/level.h"
```

Macros

- `#define BUF_SIZE 100`
- `#define RED "\x1B[31m"`
- `#define GRN "\x1B[32m"`
- `#define YEL "\x1B[33m"`
- `#define BLU "\x1B[34m"`
- `#define MAG "\x1B[35m"`
- `#define CYN "\x1B[36m"`
- `#define WHT "\x1B[37m"`
- `#define RESET "\x1B[0m"`

Functions

- `int damageCalculationPlayer (SkillInfo skill, LevelInfo player, MonsterInfo monster)`
Player's damage calculation.
- `int damageCalculationMonster (SkillInfo skill, LevelInfo player, MonsterInfo monster)`
Monster's damage calculation.
- `SkillInfo monsterAI (MonsterInfo monster, int curHP)`
Monster's behavior.
- `void campaign ()`
PVE mode.
- `char gameoverChoice ()`
Game over menu.
- `char stageoverChoice ()`
Game over menu.
- `void printUserLog (char user[], char skill[], int dmg, Type type)`
Print User log.
- `void printMonsterLog (char monster[], char skill[], int dmg, Type type)`
Print Monster log.

4.3.1 Macro Definition Documentation

4.3.1.1 BLU

```
#define BLU "\x1B[34m"
```

Print in Blue color

4.3.1.2 BUF_SIZE

```
#define BUF_SIZE 100
```

Maximum buffer size

4.3.1.3 CYN

```
#define CYN "\x1B[36m"
```

Print in Cyn color

4.3.1.4 GRN

```
#define GRN "\x1B[32m"
```

Print in Greed color

4.3.1.5 MAG

```
#define MAG "\x1B[35m"
```

Print in Mag color

4.3.1.6 RED

```
#define RED "\x1B[31m"
```

Print in Red color

4.3.1.7 RESET

```
#define RESET "\x1B[0m"
```

Print Reset

4.3.1.8 WHT

```
#define WHT "\x1B[37m"
```

Print in White color

4.3.1.9 YEL

```
#define YEL "\x1B[33m"
```

Print in Yellow color

4.3.2 Function Documentation

4.3.2.1 campaign()

```
void campaign ( )
```

PVE mode.

4.3.2.2 damageCalculationMonster()

```
int damageCalculationMonster (
    SkillInfo skill,
    LevelInfo player,
    MonsterInfo monster )
```

Monster's damage calculation.

Parameters

<i>SkillInfo</i>	Monster skill
<i>LevelInfo</i>	Player's level info
<i>MonsterInfo</i>	Monster information

4.3.2.3 damageCalculationPlayer()

```
int damageCalculationPlayer (
    SkillInfo skill,
```

```
LevelInfo player,  
MonsterInfo monster )
```

Player's damage calculation.

Parameters

<i>SkillInfo</i>	Player skill
<i>LevelInfo</i>	Player's level info
<i>MonsterInfo</i>	Monster information

4.3.2.4 gameoverChoice()

```
char gameoverChoice ( )
```

Game over menu.

Returns

Player choice

4.3.2.5 monsterAI()

```
SkillInfo monsterAI (  
    MonsterInfo monster,  
    int curHP )
```

Monster's behavior.

Parameters

<i>MonsterInfo</i>	Monster information
<i>int</i>	Monster current HP

Returns

Monster Skill

4.3.2.6 printMonsterLog()

```
void printMonsterLog (  
    char monster[],
```

```
char skill[],  
int dmg,  
Type type )
```

Print Monster log.

Parameters

<i>String</i>	Monster name
<i>String</i>	Skill name
<i>int</i>	Damage
<i>Type</i>	Skill type

4.3.2.7 printUserLog()

```
void printUserLog (  
    char user[],  
    char skill[],  
    int dmg,  
    Type type )
```

Print User log.

Parameters

<i>String</i>	Username
<i>String</i>	Skill name
<i>int</i>	Damage
<i>Type</i>	Skill type

4.3.2.8 stageoverChoice()

```
char stageoverChoice ( )
```

Game over menu.

Returns

Player choice

4.4 game/gamemaster.c File Reference

```
#include <stdio.h>  
#include <stdlib.h>  
#include "gamemaster.h"
```

Functions

- `UserNode * loadUserInfo ()`
Load UserInfo from file.

4.4.1 Function Documentation

4.4.1.1 loadUserInfo()

```
UserNode* loadUserInfo ( )
```

Load UserInfo from file.

Returns

A list of UserNode

4.5 game/gamemaster.h File Reference

```
#include "../struct/user.h"
```

Functions

- `UserNode * loadUserInfo ()`
Load UserInfo from file.

Variables

- `UserNode * root`

4.5.1 Function Documentation

4.5.1.1 loadUserInfo()

```
UserNode* loadUserInfo ( )
```

Load UserInfo from file.

Returns

A list of UserNode

4.5.2 Variable Documentation

4.5.2.1 root

```
UserNode* root
```

4.6 helper/helper.c File Reference

```
#include "helper.h"
```

Functions

- char * [userHighScoreFormat](#) (char *username, int stage)
Format highscore message.

4.6.1 Function Documentation

4.6.1.1 userHighScoreFormat()

```
char* userHighScoreFormat (  
    char * username,  
    int stage )
```

Format highscore message.

Parameters

<i>String</i>	username
<i>int</i>	user current stage

Returns

String with Format: "username : stage"

4.7 helper/helper.h File Reference

```
#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>
```

Macros

- `#define MAX_STAGE_IN_STRING 4`

Functions

- `char * userHighScoreFormat (char *username, int stage)`
Format highscore message.

4.7.1 Macro Definition Documentation

4.7.1.1 MAX_STAGE_IN_STRING

```
#define MAX_STAGE_IN_STRING 4
```

4.7.2 Function Documentation

4.7.2.1 userHighScoreFormat()

```
char* userHighScoreFormat (  
    char * username,  
    int stage )
```

Format highscore message.

Parameters

<i>String</i>	username
<i>int</i>	user current stage

Returns

String with Format: "username : stage"

4.8 helper/mysocket.c File Reference

```
#include "mysocket.h"
```

Functions

- int [getPort](#) (const char *port_argument)
Get port number from argument.
- void [die](#) (char *msg, int type)
Exit program when error encounter or get exit message.
- struct sockaddr_in [configAddress](#) (const char *ip, int port)
Config an address with ip and port.

4.8.1 Function Documentation

4.8.1.1 configAddress()

```
struct sockaddr_in configAddress (  
    const char * ip,  
    int port )
```

Config an address with ip and port.

Parameters

<i>String</i>	ip address
<i>int</i>	port number

Returns

sockaddr_in

4.8.1.2 die()

```
void die (  
    char * msg,  
    int type )
```

Exit program when error encounter or get exit message.

Parameters

<i>String</i>	message
<i>int</i>	type

4.8.1.3 getPort()

```
int getPort (
    const char * port_argument )
```

Get port number from argument.

Parameters

<i>String</i>	port number
---------------	-------------

Returns

-1 on invalid
port number on valid

4.9 helper/mysocket.h File Reference

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <string.h>
#include <ctype.h>
#include <sys/select.h>
#include <sys/time.h>
#include <errno.h>
```

Functions

- int [getPort](#) (const char *port_argument)
Get port number from argument.
- void [die](#) (char *msg, int type)
Exit program when error encounter or get exit message.
- struct sockaddr_in [configAddress](#) (const char *ip, int port)
Config an address with ip and port.

4.9.1 Function Documentation

4.9.1.1 configAddress()

```
struct sockaddr_in configAddress (
    const char * ip,
    int port )
```

Config an address with ip and port.

Parameters

<i>String</i>	ip address
<i>int</i>	port number

Returns

sockaddr_in

4.9.1.2 die()

```
void die (
    char * msg,
    int type )
```

Exit program when error encounter or get exit message.

Parameters

<i>String</i>	message
<i>int</i>	type

4.9.1.3 getPort()

```
int getPort (
    const char * port_argument )
```

Get port number from argument.

Parameters

<i>String</i>	port number
---------------	-------------

Returns

-1 on invalid
port number on valid

4.10 interface/clientfunc.c File Reference

```
#include "clientfunc.h"
#include "../game/game.h"
```

```
#include <string.h>
#include <stdlib.h>
```

Functions

- void [login](#) (int connfd)
Fill login form and send to server.
- void [registerFunc](#) (int connfd)
Fill register form and send to server.
- void [fetchPlayerData](#) (int connfd)
Get player data from server.
- void [fetchHighScore](#) (int connfd)
Get highscore list from server.
- void [updateUserInfo](#) (int connfd, [UserInfo](#) user)
Request for update user current status.
- void [playMenu](#) (int connfd)
Play menu.

Variables

- [UserInfo](#) user
- struct sockaddr_in [servaddr](#)

4.10.1 Function Documentation

4.10.1.1 [fetchHighScore\(\)](#)

```
void fetchHighScore (
    int connfd )
```

Get highscore list from server.

Parameters

<i>int</i>	connection file descriptor
------------	----------------------------

4.10.1.2 [fetchPlayerData\(\)](#)

```
void fetchPlayerData (
    int connfd )
```

Get player data from server.

Parameters

<i>int</i>	connection file descriptor
------------	----------------------------

4.10.1.3 login()

```
void login (  
    int connfd )
```

Fill login form and send to server.

Parameters

<i>int</i>	connection file descriptor
------------	----------------------------

4.10.1.4 playMenu()

```
void playMenu (  
    int connfd )
```

Play menu.

Parameters

<i>int</i>	connection file descriptor
------------	----------------------------

4.10.1.5 registerFunc()

```
void registerFunc (  
    int connfd )
```

Fill register form and send to server.

Parameters

<i>int</i>	connection file descriptor
------------	----------------------------

4.10.1.6 updateUserInfo()

```
void updateUserInfo (
    int connfd,
    UserInfo user )
```

Request for update user current status.

Parameters

<i>int</i>	connection file descriptor
------------	----------------------------

4.10.2 Variable Documentation

4.10.2.1 servaddr

```
struct sockaddr_in servaddr
```

4.10.2.2 user

```
UserInfo user
```

4.11 interface/clientfunc.h File Reference

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <string.h>
#include <ctype.h>
#include <sys/select.h>
#include <sys/time.h>
#include <errno.h>
#include "../helper/mysocket.h"
#include "../struct/user.h"
```


Macros

- `#define BUFFER_LEN 1024`

Functions

- void `login` (int connfd)
Fill login form and send to server.
- void `registerFunc` (int connfd)
Fill register form and send to server.
- void `fetchPlayerData` (int connfd)
Get player data from server.
- void `playMenu` (int connfd)
Play menu.
- void `updateUserInfo` (int connfd, `UserInfo` user)
Request for update user current status.
- void `fetchHighScore` (int connfd)
Get highscore list from server.

4.11.1 Macro Definition Documentation

4.11.1.1 BUFFER_LEN

```
#define BUFFER_LEN 1024
```

Maximum of buffer lenght

4.11.2 Function Documentation

4.11.2.1 fetchHighScore()

```
void fetchHighScore (  
    int connfd )
```

Get highscore list from server.

Parameters

<code>int</code>	connection file descriptor
------------------	----------------------------

4.11.2.2 `fetchPlayerData()`

```
void fetchPlayerData (  
    int connfd )
```

Get player data from server.

Parameters

<i>int</i>	connection file descriptor
------------	----------------------------

4.11.2.3 `login()`

```
void login (  
    int connfd )
```

Fill login form and send to server.

Parameters

<i>int</i>	connection file descriptor
------------	----------------------------

4.11.2.4 `playMenu()`

```
void playMenu (  
    int connfd )
```

Play menu.

Parameters

<i>int</i>	connection file descriptor
------------	----------------------------

4.11.2.5 `registerFunc()`

```
void registerFunc (  
    int connfd )
```

Fill register form and send to server.

Parameters

<i>int</i>	connection file descriptor
------------	----------------------------

4.11.2.6 updateUserInfo()

```
void updateUserInfo (
    int connfd,
    UserInfo user )
```

Request for update user current status.

Parameters

<i>int</i>	connection file descriptor
------------	----------------------------

4.12 interface/serverfunc.c File Reference

```
#include "serverfunc.h"
#include "../helper/helper.h"
```

Functions

- int [handleRequest](#) (int connfd, [UserNode](#) **root)
Redirect client request to right function.
- int [login](#) (int connfd, [UserNode](#) *root)
User login handle.
- int [sendPlayerInfo](#) (int connfd, [UserNode](#) *root)
Send player info to client.
- int [sendHighScore](#) (int connfd, [UserNode](#) *root)
Send users high score to client.
- int [updatePlayerInfo](#) (int connfd, [UserNode](#) **root)
Update user stat.
- int [addUser](#) (int connfd, [UserNode](#) **root)
Add new user when client register.

4.12.1 Function Documentation

4.12.1.1 addUser()

```
int addUser (
    int connfd,
    UserNode ** root )
```

Add new user when client register.

Parameters

<i>int</i>	connection file descriptor
<i>UserNode</i>	pointer to user list

Returns

1 on success
0 on error

4.12.1.2 handleRequest()

```
int handleRequest (
    int connfd,
    UserNode ** root )
```

Redirect client request to right function.

Parameters

--	--

4.12.1.3 login()

```
int login (
    int connfd,
    UserNode * root )
```

User login handle.

Parameters

<i>int</i>	connection file description
<i>UserNode</i>	user list root

Returns

1 on success
0 on error

4.12.1.4 sendHighScore()

```
int sendHighScore (
    int connfd,
    UserNode * root )
```

Send users high score to client.

Parameters

<i>int</i>	connection file description
<i>UserNode</i>	user list root

Returns

1 on success
0 on error

4.12.1.5 sendPlayerInfo()

```
int sendPlayerInfo (
    int connfd,
    UserNode * root )
```

Send player info to client.

Parameters

<i>int</i>	connection file description
<i>UserNode</i>	user list root

Returns

1 on success
0 on error

4.12.1.6 updatePlayerInfo()

```
int updatePlayerInfo (
    int connfd,
    UserNode ** root )
```

Update user stat.

Parameters

<i>int</i>	connection file descriptor
<i>UserNode*</i>	pointer to user list

Returns

- 1 on success
- 0 on error

4.13 interface/serverfunc.h File Reference

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <string.h>
#include <ctype.h>
#include <sys/select.h>
#include <sys/time.h>
#include <errno.h>
#include "../helper/mysocket.h"
#include "../struct/user.h"
```

Macros

- `#define BUFFER_LEN 1024`

Functions

- `int handleRequest (int connfd, UserNode **root)`
Redirect client request to right function.
- `int login (int connfd, UserNode *root)`
User login handle.
- `int addUser (int connfd, UserNode **root)`
Add new user when client register.
- `int sendPlayerInfo (int connfd, UserNode *root)`
Send player info to client.
- `int updatePlayerInfo (int connfd, UserNode **root)`
Update user stat.
- `int sendHighScore (int connfd, UserNode *root)`
Send users high score to client.

4.13.1 Macro Definition Documentation

4.13.1.1 BUFFER_LEN

```
#define BUFFER_LEN 1024
```

Maximum of buffer lenght

4.13.2 Function Documentation

4.13.2.1 addUser()

```
int addUser (
    int connfd,
    UserNode ** root )
```

Add new user when client register.

Parameters

<i>int</i>	connection file descriptor
<i>UserNode</i>	pointer to user list

Returns

1 on success
0 on error

4.13.2.2 handleRequest()

```
int handleRequest (
    int connfd,
    UserNode ** root )
```

Redirect client request to right function.

Parameters

--	--

4.13.2.3 login()

```
int login (
    int connfd,
    UserNode * root )
```

User login handle.

Parameters

<i>int</i>	connection file description
<i>UserNode</i>	user list root

Returns

1 on success
0 on error

4.13.2.4 sendHighScore()

```
int sendHighScore (
    int connfd,
    UserNode * root )
```

Send users high score to client.

Parameters

<i>int</i>	connection file description
<i>UserNode</i>	user list root

Returns

1 on success
0 on error

4.13.2.5 sendPlayerInfo()

```
int sendPlayerInfo (
    int connfd,
    UserNode * root )
```

Send player info to client.

Parameters

<i>int</i>	connection file description
<i>UserNode</i>	user list root

Returns

1 on success
0 on error

4.13.2.6 updatePlayerInfo()

```
int updatePlayerInfo (
    int connfd,
    UserNode ** root )
```

Update user stat.

Parameters

<i>int</i>	connection file descriptor
<i>UserNode*</i>	pointer to user list

Returns

1 on success
0 on error

4.14 server.c File Reference

```
#include "helper/mysocket.h"
#include "game/gamemaster.h"
#include "interface/serverfunc.h"
```

Macros

- #define `MAX_CLIENT` 30

Functions

- int `main` (int argc, char const *argv[])

Variables

- `UserNode * root = NULL`

4.14.1 Macro Definition Documentation

4.14.1.1 MAX_CLIENT

```
#define MAX_CLIENT 30
```

4.14.2 Function Documentation

4.14.2.1 main()

```
int main (
    int argc,
    char const * argv[] )
```

4.14.3 Variable Documentation

4.14.3.1 root

```
UserNode* root = NULL
```

4.15 struct/level.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "level.h"
#include "skill.h"
```

Functions

- void `loadLevelInfo ()`
Load LevelInfo from file.

Variables

- [SkillInfo skills](#) []
- [LevelInfo levels](#) [MAX_LEVEL]

4.15.1 Function Documentation

4.15.1.1 loadLevelInfo()

```
void loadLevelInfo ( )
```

Load LevelInfo from file.

4.15.2 Variable Documentation

4.15.2.1 levels

```
LevelInfo levels[MAX_LEVEL]
```

4.15.2.2 skills

```
SkillInfo skills[]
```

4.16 struct/level.h File Reference

```
#include "skill.h"
```

Classes

- struct [LevelInformation](#)
LevelInformation Structure.

Macros

- #define [MAX_LEVEL](#) 5

Typedefs

- typedef struct [LevelInformation](#) [LevelInfo](#)
[LevelInformation](#) Structure.

Functions

- void [loadLevelInfo](#) ()
Load LevelInfo from file.

4.16.1 Macro Definition Documentation

4.16.1.1 MAX_LEVEL

```
#define MAX_LEVEL 5
```

4.16.2 Typedef Documentation

4.16.2.1 LevelInfo

```
typedef struct LevelInformation LevelInfo
```

[LevelInformation](#) Structure.

Structure to store Player Level's Information

4.16.3 Function Documentation

4.16.3.1 loadLevelInfo()

```
void loadLevelInfo ( )
```

Load LevelInfo from file.

4.17 struct/monster.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "monster.h"
#include "skill.h"
```

Functions

- void [loadMonsterInfo](#) ()
Load Monster Info from file.

Variables

- [SkillInfo](#) [skills](#) []
- [MonsterInfo](#) [monsters](#) [[MAX_MONSTER](#)]

4.17.1 Function Documentation

4.17.1.1 [loadMonsterInfo](#)()

```
void loadMonsterInfo ( )
```

Load Monster Info from file.

4.17.2 Variable Documentation

4.17.2.1 [monsters](#)

```
MonsterInfo monsters [MAX\_MONSTER]
```

4.17.2.2 [skills](#)

```
SkillInfo skills []
```

4.18 struct/monster.h File Reference

```
#include "skill.h"
```

Classes

- struct [MonsterInformation](#)
MonsterInformation structure.

Macros

- #define [MAX_MONSTER](#) 2

Typedefs

- typedef struct [MonsterInformation](#) [MonsterInfo](#)
MonsterInformation structure.

Functions

- void [loadMonsterInfo](#) ()
Load Monster Info from file.

4.18.1 Macro Definition Documentation

4.18.1.1 MAX_MONSTER

```
#define MAX_MONSTER 2
```

Maximum number of Monster

4.18.2 Typedef Documentation

4.18.2.1 MonsterInfo

```
typedef struct MonsterInformation MonsterInfo
```

[MonsterInformation](#) structure.

Structure to store monster's information

4.18.3 Function Documentation

4.18.3.1 loadMonsterInfo()

```
void loadMonsterInfo ( )
```

Load Monster Info from file.

4.19 struct/skill.c File Reference

```
#include "skill.h"
```

Variables

- [SkillInfo skills](#) [[SKILL_COUNT](#)]

4.19.1 Variable Documentation

4.19.1.1 skills

```
SkillInfo skills[SKILL_COUNT]
```

Initial value:

```
= {
    {S_EMBER, "Ember", 20, 5, T_FIRE},
    {S_VINEWHIP, "Vine Whip", 20, 5, T_GRASS},
    {S_WATERGUN, "Water Gun", 20, 5, T_WATER},
    {S_HEADBUTT, "Headbutt", 30, 7, T_NORMAL},
    {S_SLASH, "Slash", 35, 8, T_NORMAL},
    {S_FLAMEWHEEL, "Flame Wheel", 30, 10, T_FIRE},
    {S_MAGICALLEAF, "Magical Leaf", 30, 10, T_GRASS},
    {S_WATERPULSE, "Water Pulse", 30, 10, T_WATER},
    {S_FLAMETHROWER, "FlameThrower", 35, 15, T_FIRE},
    {S_LEAFBLADE, "Leaf Blade", 35, 15, T_GRASS},
    {S_SURF, "Surf", 35, 15, T_WATER},
    {S_MEGAPUNCH, "Mega Punch", 40, 15, T_NORMAL},
    {S_FIREBLAST, "Fire Blast", 40, 20, T_FIRE},
    {S_SOLARBEAM, "Solar Beam", 40, 20, T_GRASS},
    {S_HYDROPUMP, "Hydro Pump", 40, 20, T_WATER},
    {S_MEGAKICK, "Mega Kick", 45, 20, T_NORMAL},
    {S_BLASTBURN, "Blast Burn", 45, 25, T_FIRE},
    {S_FRENZYPLANT, "Frenzy Plant", 45, 25, T_GRASS},
    {S_HYDROCANNON, "Hydro Cannon", 45, 25, T_WATER},
    {S_HYPERBEAM, "Hyper Beam", 45, 25, T_NORMAL},
    {ES_TACKLE, "Tackle", 10, 0, T_NORMAL},
    {ES_ACID, "Acid", 15, 0, T_NORMAL},
    {ES_SLUDGEBOOMB, "Sludge Bomb", 20, 0, T_NORMAL},
    {ES_GUNKSHOT, "Gunk Shot", 25, 0, T_NORMAL},
    {ES_SCRATCH, "Scratch", 10, 0, T_NORMAL},
    {ES_THRUST, "Thrust", 15, 0, T_NORMAL},
    {ES_POTION, "Potion", 5, 0, T_HEAL},
    {ES_GOBLINPUNCH, "Goblin Punch", 25, 0, T_NORMAL},
    {S_NORMALATTACK, "Normal Attack", 10, 0, T_NORMAL}}
```

4.20 struct/skill.h File Reference

Classes

- struct [SkillInformation](#)
[SkillInformation](#) Structure.

Macros

- #define [STRING_LEN](#) 20
- #define [SKILL_COUNT](#) 29

Typedefs

- typedef struct [SkillInformation](#) [SkillInfo](#)
[SkillInformation](#) Structure.

Enumerations

-


```
enum Skill {
    S_EMBER, S_VINEWHIP, S_WATERGUN, S_HEADBUTT,
    S_SLASH, S_FLAMEWHEEL, S_MAGICALLEAF, S_WATERPULSE,
    S_FLAMETHROWER, S_LEAFBLADE, S_SURF, S_MEGAPUNCH,
    S_FIREBLAST, S_SOLARBEAM, S_HYDROPUMP, S_MEGAKICK,
    S_BLASTBURN, S_FRENZYPLANT, S_HYDROCANNON, S_HYPERBEAM,
    ES_TACKLE, ES_ACID, ES_SLUDGEBOOM, ES_GUNKSHOT,
    ES_SCRATCH, ES_THRUST, ES_POTION, ES_GOBLINPUNCH,
    S_NORMALATTACK }

```

Skill enum.

- enum Type {


```
T_FIRE, T_GRASS, T_WATER, T_NORMAL,
T_HEAL }
```

Type enum.

4.20.1 Macro Definition Documentation

4.20.1.1 SKILL_COUNT

```
#define SKILL_COUNT 29
```

Maximum number of skill

4.20.1.2 STRING_LEN

```
#define STRING_LEN 20
```

Maximum number of character

4.20.2 Typedef Documentation

4.20.2.1 SkillInfo

```
typedef struct SkillInformation SkillInfo
```

[SkillInformation](#) Structure.

Structure to store Skill's Information

4.20.3 Enumeration Type Documentation

4.20.3.1 Skill

```
enum Skill
```

Skill enum.

List of skill

Enumerator

S_EMBER	
S_VINEWHIP	
S_WATERGUN	
S_HEADBUTT	
S_SLASH	
S_FLAMEWHEEL	
S_MAGICALLEAF	
S_WATERPULSE	
S_FLAMETHROWER	
S_LEAFBLADE	
S_SURF	
S_MEGAPUNCH	
S_FIREBLAST	
S_SOLARBEAM	
S_HYDROPUMP	
S_MEGAKICK	
S_BLASTBURN	
S_FRENZYPLANT	
S_HYDROCANNON	
S_HYPERBEAM	
ES_TACKLE	
ES_ACID	
ES_SLUDGEBOOM	
ES_GUNKSHOT	
ES_SCRATCH	
ES_THRUST	
ES_POTION	
ES_GOBLINPUNCH	
S_NORMALATTACK	

4.20.3.2 Type

enum [Type](#)

Type enum.

List of Type

Enumerator

T_FIRE	
T_GRASS	
T_WATER	
T_NORMAL	
T_HEAL	

4.21 struct/stage.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "stage.h"
#include "monster.h"
```

Functions

- void [loadStageInfo](#) ()
Load Stage Info from file.

Variables

- [MonsterInfo](#) [monsters](#) []
- [StageInfo](#) [stages](#) [[MAX_STAGE](#)]

4.21.1 Function Documentation

4.21.1.1 loadStageInfo()

```
void loadStageInfo ( )
```

Load Stage Info from file.

4.21.2 Variable Documentation

4.21.2.1 monsters

```
MonsterInfo monsters []
```

4.21.2.2 stages

```
StageInfo stages [MAX\_STAGE]
```

4.22 struct/stage.h File Reference

```
#include "monster.h"
```

Classes

- struct [StageInformation](#)
StageInformation structure.

Macros

- #define [MAX_STAGE](#) 2

Typedefs

- typedef struct [StageInformation](#) [StageInfo](#)
StageInformation structure.

Functions

- void [loadStageInfo](#) ()
Load Stage Info from file.

4.22.1 Macro Definition Documentation

4.22.1.1 MAX_STAGE

```
#define MAX_STAGE 2
```

Maximum number of stage

4.22.2 Typedef Documentation

4.22.2.1 StageInfo

```
typedef struct StageInformation StageInfo
```

[StageInformation](#) structure.

Structure to store stage's information

4.22.3 Function Documentation

4.22.3.1 loadStageInfo()

```
void loadStageInfo ( )
```

Load Stage Info from file.

4.23 struct/user.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include "user.h"
```

Functions

- `UserNode * findUser (UserNode *root, char *username)`
Find an User from UserNode list.
- `UserNode * makeNewNode (UserInfo data)`
Create new UserNode with UserInfo.
- `UserNode * insertNode (UserNode *root, UserNode *new)`
Insert a node to UserNode list.
- `void freeList (UserNode *root)`
Free an UserNode list.
- `void printList (UserNode *root)`
Print an UserNode list.
- `UserNode * sortUserlist (UserNode *root)`
Sort a node to UserNode list.
- `UserInfo initUserInfo (char *username, char *password)`
*Initial an *UserInformation* from username and password.*
- `void writeUserData (UserNode *root)`
Write an UserNode list to file.

4.23.1 Function Documentation

4.23.1.1 findUser()

```
UserNode* findUser (
    UserNode * root,
    char * username )
```

Find an User from UserNode list.

Parameters

<i>UserNode</i>	Root of an UserNode list
<i>String</i>	Username to find

Returns

An UserNode on found
NULL on not found

4.23.1.2 freeList()

```
void freeList (
    UserNode * root )
```

Free an UserNode list.

Parameters

<i>UserNode</i>	Root of an UserNode list
-----------------	--------------------------

4.23.1.3 initUserInfo()

```
UserInfo initUserInfo (
    char * username,
    char * password )
```

Initial an [UserInformation](#) from username and password.

Parameters

<i>String</i>	username
<i>String</i>	password

Returns

new UserInfo

4.23.1.4 insertNode()

```
UserNode* insertNode (
    UserNode * root,
    UserNode * new )
```

Insert a node to UserNode list.

Parameters

<i>UserNode</i>	Root of an UserNode list
<i>UserNode</i>	Node to insert

Returns

Root of an UserNode list with inserted node

4.23.1.5 makeNewNode()

```
UserNode* makeNewNode (
    UserInfo data )
```

Create new UserNode with UserInfo.

Parameters

<i>UserInfo</i>	Data for UserNode
-----------------	-------------------

Returns

A node created with provided data

4.23.1.6 printList()

```
void printList (
    UserNode * root )
```

Print an UserNode list.

Parameters

<i>UserNode</i>	Root of an UserNode list
-----------------	--------------------------

4.23.1.7 sortUserlist()

```
UserNode* sortUserlist (
    UserNode * root )
```

Sort a node to UserNode list.

Parameters

<i>UserNode</i>	Root of an UserNode list
-----------------	--------------------------

Returns

Root of an UserNode list with sorted node

4.23.1.8 writeUserData()

```
void writeUserData (
    UserNode * root )
```

Write an UserNode list to file.

Parameters

<i>UserNode</i>	Root of an UserNode list
-----------------	--------------------------

4.24 struct/user.h File Reference

File containing User structure and usage.

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

Classes

- struct [UserInformation](#)
UserInformation structure.
- struct [UserInfoNode](#)
UserInfoNode structure.

Macros

- #define [STRING_LEN](#) 20

Typedefs

- typedef struct [UserInformation](#) [UserInfo](#)
UserInformation structure.
- typedef struct [UserInfoNode](#) [UserNode](#)
UserInfoNode structure.

Functions

- `UserNode * makeNewNode (UserInfo data)`
Create new UserNode with UserInfo.
- `UserNode * insertNode (UserNode *root, UserNode *new)`
Insert a node to UserNode list.
- `void freeList (UserNode *root)`
Free an UserNode list.
- `void printList (UserNode *root)`
Print an UserNode list.
- `UserNode * findUser (UserNode *root, char *username)`
Find an User from UserNode list.
- `UserNode * sortUserlist (UserNode *root)`
Sort a node to UserNode list.
- `UserInfo initUserInfo (char *username, char *password)`
*Initial an *UserInfo* from username and password.*
- `void writeUserData (UserNode *root)`
Write an UserNode list to file.

4.24.1 Detailed Description

File containing User structure and usage.

4.24.2 Macro Definition Documentation

4.24.2.1 STRING_LEN

```
#define STRING_LEN 20
```

Maximum number of character

4.24.3 Typedef Documentation

4.24.3.1 UserInfo

```
typedef struct UserInfo UserInfo
```

UserInfo structure.

Structure to store user's information

4.24.3.2 UserNode

```
typedef struct UserInfoNode UserNode
```

[UserInfoNode](#) structure.

Structure to store user's information as linked list

4.24.4 Function Documentation

4.24.4.1 findUser()

```
UserNode* findUser (  
    UserNode * root,  
    char * username )
```

Find an User from UserNode list.

Parameters

<i>UserNode</i>	Root of an UserNode list
<i>String</i>	Username to find

Returns

An UserNode on found
NULL on not found

4.24.4.2 freeList()

```
void freeList (  
    UserNode * root )
```

Free an UserNode list.

Parameters

<i>UserNode</i>	Root of an UserNode list
-----------------	--------------------------

4.24.4.3 initUserInfo()

```
UserInfo initUserInfo (
```

```
char * username,  
char * password )
```

Initial an [UserInfo](#) from username and password.

Parameters

<i>String</i>	username
<i>String</i>	password

Returns

new UserInfo

4.24.4.4 insertNode()

```
UserNode* insertNode (  
    UserNode * root,  
    UserNode * new )
```

Insert a node to UserNode list.

Parameters

<i>UserNode</i>	Root of an UserNode list
<i>UserNode</i>	Node to insert

Returns

Root of an UserNode list with inserted node

4.24.4.5 makeNewNode()

```
UserNode* makeNewNode (  
    UserInfo data )
```

Create new UserNode with UserInfo.

Parameters

<i>UserInfo</i>	Data for UserNode
-----------------	-------------------

Returns

A node created with provided data

4.24.4.6 printList()

```
void printList (
    UserNode * root )
```

Print an UserNode list.

Parameters

<i>UserNode</i>	Root of an UserNode list
-----------------	--------------------------

4.24.4.7 sortUserlist()

```
UserNode* sortUserlist (
    UserNode * root )
```

Sort a node to UserNode list.

Parameters

<i>UserNode</i>	Root of an UserNode list
-----------------	--------------------------

Returns

Root of an UserNode list with sorted node

4.24.4.8 writeUserData()

```
void writeUserData (
    UserNode * root )
```

Write an UserNode list to file.

Parameters

<i>UserNode</i>	Root of an UserNode list
-----------------	--------------------------

Index

- addUser
 - serverfunc.c, [39](#)
 - serverfunc.h, [43](#)
- atk
 - LevelInformation, [5](#)
 - MonsterInformation, [7](#)
- BLU
 - game.h, [24](#)
- BUF_SIZE
 - game.h, [24](#)
- BUFFER_LEN
 - client.c, [16](#)
 - clientfunc.h, [37](#)
 - serverfunc.h, [43](#)
- CYN
 - game.h, [24](#)
- campaign
 - game.c, [18](#)
 - game.h, [25](#)
- ch
 - client.c, [16](#)
- client.c, [15](#)
 - BUFFER_LEN, [16](#)
 - ch, [16](#)
 - main, [16](#)
 - servaddr, [16](#)
 - sockfd, [16](#)
 - temp, [16](#)
 - user, [16](#)
- clientfunc.c
 - fetchHighScore, [34](#)
 - fetchPlayerData, [34](#)
 - login, [35](#)
 - playMenu, [35](#)
 - registerFunc, [35](#)
 - servaddr, [36](#)
 - updateUserInfo, [35](#)
 - user, [36](#)
- clientfunc.h
 - BUFFER_LEN, [37](#)
 - fetchHighScore, [37](#)
 - fetchPlayerData, [37](#)
 - login, [38](#)
 - playMenu, [38](#)
 - registerFunc, [38](#)
 - updateUserInfo, [39](#)
- configAddress
 - mysocket.c, [31](#)
- mysocket.h, [32](#)
- count
 - LevelInformation, [5](#)
- curDmg
 - game.c, [21](#)
- curExp
 - UserInformation, [12](#)
- curHP
 - UserInformation, [12](#)
- curMP
 - UserInformation, [12](#)
- damageCalculationMonster
 - game.c, [18](#)
 - game.h, [25](#)
- damageCalculationPlayer
 - game.c, [18](#)
 - game.h, [25](#)
- def
 - LevelInformation, [6](#)
 - MonsterInformation, [7](#)
- die
 - mysocket.c, [31](#)
 - mysocket.h, [33](#)
- dmg
 - SkillInformation, [9](#)
- exp
 - MonsterInformation, [7](#)
- fetchHighScore
 - clientfunc.c, [34](#)
 - clientfunc.h, [37](#)
- fetchPlayerData
 - clientfunc.c, [34](#)
 - clientfunc.h, [37](#)
- findUser
 - user.c, [57](#)
 - user.h, [63](#)
- freeList
 - user.c, [58](#)
 - user.h, [63](#)
- GRN
 - game.h, [24](#)
- game.c
 - campaign, [18](#)
 - curDmg, [21](#)
 - damageCalculationMonster, [18](#)
 - damageCalculationPlayer, [18](#)

- gameoverChoice, 19
- levels, 21
- monsterAI, 19
- monsterCurHP, 21
- monsterCurMP, 21
- monsterSkill, 21
- monsters, 21
- printMonsterLog, 19
- printUserLog, 20
- skills, 21
- sockfd, 21
- stageoverChoice, 20
- stages, 22
- user, 22
- userCurExp, 22
- userCurHP, 22
- userCurLevel, 22
- userCurMP, 22
- userCurStage, 22
- userStageHP, 22
- userStageMP, 23
- game.h
 - BLU, 24
 - BUF_SIZE, 24
 - CYN, 24
 - campaign, 25
 - damageCalculationMonster, 25
 - damageCalculationPlayer, 25
 - GRN, 24
 - gameoverChoice, 26
 - MAG, 24
 - monsterAI, 26
 - printMonsterLog, 26
 - printUserLog, 27
 - RESET, 24
 - RED, 24
 - stageoverChoice, 27
 - WHT, 24
 - YEL, 25
- game/game.c, 17
- game/game.h, 23
- game/gamemaster.c, 27
- game/gamemaster.h, 28
- gamemaster.c
 - loadUserInfo, 28
- gamemaster.h
 - loadUserInfo, 28
 - root, 29
- gameoverChoice
 - game.c, 19
 - game.h, 26
- getPort
 - mysocket.c, 31
 - mysocket.h, 33
- handleRequest
 - serverfunc.c, 40
 - serverfunc.h, 43
- helper.c
 - userHighScoreFormat, 29
- helper.h
 - MAX_STAGE_IN_STRING, 30
 - userHighScoreFormat, 30
- helper/helper.c, 29
- helper/helper.h, 29
- helper/mysocket.c, 30
- helper/mysocket.h, 32
- hp
 - LevelInformation, 6
 - MonsterInformation, 7
- initUserInfo
 - user.c, 58
 - user.h, 63
- insertNode
 - user.c, 58
 - user.h, 64
- interface/clientfunc.c, 33
- interface/clientfunc.h, 36
- interface/serverfunc.c, 39
- interface/serverfunc.h, 42
- level
 - LevelInformation, 6
 - UserInformation, 12
- level.c
 - levels, 47
 - loadLevelInfo, 47
 - skills, 47
- level.h
 - LevelInfo, 48
 - loadLevelInfo, 48
 - MAX_LEVEL, 48
- LevelInfo
 - level.h, 48
- LevelInformation, 5
 - atk, 5
 - count, 5
 - def, 6
 - hp, 6
 - level, 6
 - maxexp, 6
 - mp, 6
 - skills, 6
- levels
 - game.c, 21
 - level.c, 47
- loadLevelInfo
 - level.c, 47
 - level.h, 48
- loadMonsterInfo
 - monster.c, 49
 - monster.h, 51
- loadStageInfo
 - stage.c, 55
 - stage.h, 57
- loadUserInfo
 - gamemaster.c, 28

- gamemaster.h, 28
- loggedin
 - UserInfoNode, 11
- login
 - clientfunc.c, 35
 - clientfunc.h, 38
 - serverfunc.c, 40
 - serverfunc.h, 44
- MAX_CLIENT
 - server.c, 46
- MAX_LEVEL
 - level.h, 48
- MAX_MONSTER
 - monster.h, 50
- MAX_STAGE_IN_STRING
 - helper.h, 30
- MAX_STAGE
 - stage.h, 56
- MAG
 - game.h, 24
- main
 - client.c, 16
 - server.c, 46
- makeNewNode
 - user.c, 60
 - user.h, 64
- maxexp
 - LevelInformation, 6
- monster.c
 - loadMonsterInfo, 49
 - monsters, 49
 - skills, 49
- monster.h
 - loadMonsterInfo, 51
 - MAX_MONSTER, 50
 - MonsterInfo, 50
- monsterAI
 - game.c, 19
 - game.h, 26
- monsterCurHP
 - game.c, 21
- monsterCurMP
 - game.c, 21
- MonsterInfo
 - monster.h, 50
- MonsterInformation, 7
 - atk, 7
 - def, 7
 - exp, 7
 - hp, 7
 - mp, 8
 - name, 8
 - skills, 8
 - type, 8
- monsterSkill
 - game.c, 21
- monsters
 - game.c, 21
- monster.c, 49
- stage.c, 55
- StageInformation, 10
- mp
 - LevelInformation, 6
 - MonsterInformation, 8
- mpcost
 - SkillInformation, 9
- mysocket.c
 - configAddress, 31
 - die, 31
 - getPort, 31
- mysocket.h
 - configAddress, 32
 - die, 33
 - getPort, 33
- name
 - MonsterInformation, 8
 - SkillInformation, 9
- next
 - UserInfoNode, 11
- number
 - StageInformation, 10
- password
 - UserInfoInformation, 12
- playMenu
 - clientfunc.c, 35
 - clientfunc.h, 38
- printList
 - user.c, 60
 - user.h, 65
- printMonsterLog
 - game.c, 19
 - game.h, 26
- printUserLog
 - game.c, 20
 - game.h, 27
- RESET
 - game.h, 24
- RED
 - game.h, 24
- registerFunc
 - clientfunc.c, 35
 - clientfunc.h, 38
- root
 - gamemaster.h, 29
 - server.c, 46
- SKILL_COUNT
 - skill.h, 53
- STRING_LEN
 - skill.h, 53
 - user.h, 62
- sendHighScore
 - serverfunc.c, 40
 - serverfunc.h, 44

- sendPlayerInfo
 - serverfunc.c, [41](#)
 - serverfunc.h, [44](#)
- servaddr
 - client.c, [16](#)
 - clientfunc.c, [36](#)
- server.c, [45](#)
 - MAX_CLIENT, [46](#)
 - main, [46](#)
 - root, [46](#)
- serverfunc.c
 - addUser, [39](#)
 - handleRequest, [40](#)
 - login, [40](#)
 - sendHighScore, [40](#)
 - sendPlayerInfo, [41](#)
 - updatePlayerInfo, [41](#)
- serverfunc.h
 - addUser, [43](#)
 - BUFFER_LEN, [43](#)
 - handleRequest, [43](#)
 - login, [44](#)
 - sendHighScore, [44](#)
 - sendPlayerInfo, [44](#)
 - updatePlayerInfo, [45](#)
- Skill
 - skill.h, [53](#)
- skill.c
 - skills, [51](#)
- skill.h
 - SKILL_COUNT, [53](#)
 - STRING_LEN, [53](#)
 - Skill, [53](#)
 - SkillInfo, [53](#)
 - Type, [54](#)
- SkillInfo
 - skill.h, [53](#)
- SkillInformation, [8](#)
 - dmg, [9](#)
 - mpcost, [9](#)
 - name, [9](#)
 - skilltype, [9](#)
 - type, [9](#)
- skills
 - game.c, [21](#)
 - level.c, [47](#)
 - LevelInformation, [6](#)
 - monster.c, [49](#)
 - MonsterInformation, [8](#)
 - skill.c, [51](#)
- skilltype
 - SkillInformation, [9](#)
- sockfd
 - client.c, [16](#)
 - game.c, [21](#)
- sortUserlist
 - user.c, [60](#)
 - user.h, [65](#)
- stage
 - UserInformation, [13](#)
- stage.c
 - loadStageInfo, [55](#)
 - monsters, [55](#)
 - stages, [55](#)
- stage.h
 - loadStageInfo, [57](#)
 - MAX_STAGE, [56](#)
 - StageInfo, [56](#)
- StageInfo
 - stage.h, [56](#)
- StageInformation, [10](#)
 - monsters, [10](#)
 - number, [10](#)
- stageoverChoice
 - game.c, [20](#)
 - game.h, [27](#)
- stages
 - game.c, [22](#)
 - stage.c, [55](#)
- struct/level.c, [46](#)
- struct/level.h, [47](#)
- struct/monster.c, [49](#)
- struct/monster.h, [50](#)
- struct/skill.c, [51](#)
- struct/skill.h, [52](#)
- struct/stage.c, [55](#)
- struct/stage.h, [56](#)
- struct/user.c, [57](#)
- struct/user.h, [61](#)
- temp
 - client.c, [16](#)
- Type
 - skill.h, [54](#)
- type
 - MonsterInformation, [8](#)
 - SkillInformation, [9](#)
- updatePlayerInfo
 - serverfunc.c, [41](#)
 - serverfunc.h, [45](#)
- updateUserInfo
 - clientfunc.c, [35](#)
 - clientfunc.h, [39](#)
- user
 - client.c, [16](#)
 - clientfunc.c, [36](#)
 - game.c, [22](#)
 - UserInfoNode, [11](#)
- user.c
 - findUser, [57](#)
 - freeList, [58](#)
 - initUserInfo, [58](#)
 - insertNode, [58](#)
 - makeNewNode, [60](#)
 - printList, [60](#)
 - sortUserlist, [60](#)

- writeUserData, [61](#)
- user.h
 - findUser, [63](#)
 - freeList, [63](#)
 - initUserInfo, [63](#)
 - insertNode, [64](#)
 - makeNewNode, [64](#)
 - printList, [65](#)
 - STRING_LEN, [62](#)
 - sortUserlist, [65](#)
 - UserInfo, [62](#)
 - UserNode, [62](#)
 - writeUserData, [65](#)
- userCurExp
 - game.c, [22](#)
- userCurHP
 - game.c, [22](#)
- userCurLevel
 - game.c, [22](#)
- userCurMP
 - game.c, [22](#)
- userCurStage
 - game.c, [22](#)
- userHighScoreFormat
 - helper.c, [29](#)
 - helper.h, [30](#)
- UserInfo
 - user.h, [62](#)
- UserInfoNode, [10](#)
 - loggedin, [11](#)
 - next, [11](#)
 - user, [11](#)
- UserInformation, [11](#)
 - curExp, [12](#)
 - curHP, [12](#)
 - curMP, [12](#)
 - level, [12](#)
 - password, [12](#)
 - stage, [13](#)
 - username, [13](#)
- UserNode
 - user.h, [62](#)
- userStageHP
 - game.c, [22](#)
- userStageMP
 - game.c, [23](#)
- username
 - UserInformation, [13](#)
- WHT
 - game.h, [24](#)
- writeUserData
 - user.c, [61](#)
 - user.h, [65](#)
- YEL
 - game.h, [25](#)